REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-99-

0059

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12/31/98	Final Technical 9/28/95-9/30/98

4. TITLE AND SUBTITLE
"Fast Multipole Methods for Electromagnetic Circuit Computations"

5. FUNDING NUMBERS
F49620-95-C-0075

~~N/A~~

6. AUTHOR(S)
Vladimir Rokhlin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Fast Mathematical Algorithms & Hardware Corporation
1020 Sherman Avenue
Hamden, CT 06514-1337

8. PERFORMING ORGANIZATION REPORT NUMBER
3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
USAF, AFMC
Air Force Office of Scientific Research
110 Duncan Avenue, Suite B115
Bolling AFB, DC 20332-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
"Distribution Statement A. Approved for public release; Distribution is unlimited."

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
During the existence of the project, we designed Fast Multipole algorithms for the solution of the Laplace and Helmholz equations, the latter predominatel in the low-frequency regime. The principal analytical apparatus used were the new diagonal forms of translation operators for the Laplace and Helmholtz equation; such diagooal forms in turn required the use of Generalized Gaussian Quadratures, which we have also designed. In addition, we redesigned the logical structure of such algorithms, drasticall reducing their memory requirements. At the present time, extremely efficient Fast Multipole Methods for the Laplace equation in three dimensions are in existence; the construction of such schemes for the Helmholtz equation in the low-frequency regime has been reduced to a purely programming task. Six publications reporting this work have appeared (or been accepted) in refereed journals; two publications have been released in the report form, and are about to be submitted to refereed journals; three publications are in preparation

14. SUBJECT TERMS
Laplace and Helmholz equations

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
~~N/A~~ UNCLASSIFIED	~~N/A~~ UNCLASSIFIED	~~N/A~~ UNCLASSIFIED	UL ~~N/A~~

FAST MULTIPOLE METHODS FOR ELECTROMAGNETIC CIRCUIT COMPUTATIONS

Final Technical Report
December 31, 1998

Dr. Vladimir Rokhlin
Fast Mathematical Algorithms & Hardware Corporation
1020 Sherman Avenue
Hamden, CT 06514
(203) 248-8212

This research was performed under Contract #: F49620-95-C-0075

1. Final Report for Years 1995-1998

The purpose of this project has been to develop Laplace and Low-Frequency Helmholtz Fast Multipole Algorithms. These are fundamental building blocks (and computational logjams) for circuit simulation. At the inception of the project, we felt that the following steps would have to be undertaken to assure its eventual success.

1. Design the improved diagonal forms for the translation operators for the Laplace and low-frequency Helmholtz equations.

2. Design improved algorithms for the construction of Generalized Gaussian Quadratures required by the diagonal forms of the preceding paragraph.

3. Design and implement the improved FMM schemes for the Laplace and low-frequency Helmholtz equation, first in two and later in three dimensions.

4. Design the numerical machinery for the interpolation and filtering of band-limited functions on the sphere, required by advanced versions of the Helmholtz FMM.

5. Implement the final versions of the FMM for the low-frequency Helmholtz equation, both as a stand-alone device in circuit modeling and related areas, and as the sub-wavelength part of a high-frequency modeling code.

The project has been in existence for 3 years (as per our proposal), and the first four of the above five steps have been entirely completed; results of this work are reported in the publications [4], [1], [3], [2], [9], [11], [10], [12]. Of these, [4], [3], [9], [11], [10], [12] have been either published or accepted for publication; since [1], [2] have not been submitted for publication at this time, their copies are attached.

The fifth step has been in part completed at FMAH, in part by our collaborators at Boeing, and in part it is being completed at the present. Three publications are in preparation reporting this step. The following personnel have been involved in this project: R. Coifman, M. Goldberg, L. Greengard, T. Hagstrom, M. Israeli, V. Rokhlin, J. Stromberg.

Below is a description of our effort year-by-year.

Year 1

1. We started our work with the design of an improved version of the Fast Multipole Method for the Laplace equation in two dimensions. In addition to having its own applications, this scheme was viewed as a stepping stone to three-dimensional problems, first for the Laplace and later for the low-frequency Helmholtz equation. This work was completed during the first year of the project, and the paper reporting it recently appeared in the SISC (see [4]).

2. A preliminary (non-adaptive) version of the improved FMM for the Laplace equation in three dimensions was designed and implemented. The algorithm uses diagonal forms of translation operators based on the Generalized Gaussian Quadratures, designed by us specifically for this purpose (and to be used in similar situations that might arise in the future). At this stage, the FMM became an effective tool for high-precision potential calculations in three dimensions. On the other hand, the scheme implemented at this time was not adaptive, and its memory requirements were deemed excessive. The algorithm is described in [3]. As often happens, by the time we finished [3], we had constructed several additional improvements to the scheme. It was decided to incorporate these improvements in the adaptive version of the algorithm; the paper [3] does not contain them.

3. The improvements in the design of Fast Multipole algorithms described in the preceding two paragraphs were made possible by the existence of a class of diagonal representations of translation operators (for Helmholtz, Laplace, and Yukawa equations), ultimately based on the Generalized Gaussian Quadratures. The latter have been known for a very long time (the oldest references we are aware of are Markov's papers at the end of the last century). On the other hand, the classical proofs do not provide any mechanism for the numerical construction of such schemes. Thus, we launched an effort to construct an algorithm for the design of such quadratures. The effort was successful, and is reported in [10]; at that time, the obtained quadratures were virtually optimal for the Laplace equation, in both two and three dimensions. The Helmholtz and Yukawa equations required additional quadrature work.

4. "Fast" schemes for the application of discretized integral operators are a critical element of an effective circuit simulation environment. Another critical element of such an environment

is our ability to discretize the integral equations effectively. Previously, we had neglected this aspect of the problem; during the first year of the project, we started a systematic investigation of relevant discretization schemes and associated quadrature formulae.

Year 2

1. During the first year of the project, we had constructed a preliminary (non-adaptive) version of the FMM for the Laplace equation in three dimensions. While the scheme had satisfactory CPU timings for all accuracies of practical importance, its memory requirements were deemed excessive, and we felt that the design of translation operators we used could be improved. Thus, a significant amount of effort was contributed to the design of an improved FMM for the Laplace equation (principally, in three dimensions) based on improved representations of the potential; we have also designed a reformulation of the algorithm with dramatically reduced memory requirements. The resulting scheme is about twice faster than the one produced during the first year of the project, and its memory requirements were reduced by better than a factor of 5. The report describing the scheme was delayed, and appeared only in 1998 (see [1]).

2. An outstanding problem in the design of Fast Multipole Methods for the Helmholtz equation has been the expense associated with filtering and interpolation of functions on the sphere. For the FMM in two dimensions, the relevant procedure is easily performed on the circle via the FFT at a nearly optimal cost; the same operation on the sphere can not be implemented via standard algebraic procedures. During the second year of the project, we observed that there exists a version of the FMM in one dimension that performs the interpolation and filtering on the sphere in (asymptotically) optimal time; it should be mentioned that it is a modification of an earlier scheme by B. Alpert and R. Jakob-Chien (see [5]). The algorithm was implemented and made available to the Boeing and Hughes groups. Reports describing our new design of the FMM in one dimension and on the application of such schemes to filtering and interpolation on the sphere are attached (see [11, 12]).

3. One of principal purposes of this project has been the design of a version of the FMM for the Helmholtz equation at low frequencies, and of a scheme combining such a version with the high-frequency FMM; in that sense, an efficient scheme for the Laplace equation is a preliminary

step. We derived the integral representations analogous to those used in the recent FMM scheme for the Laplace equation, and tested it with a preliminary version of the quadratures. A report [9] describing these developments is attached.

Year 3

1. We have concluded the design of special-purpose quadratures to be used for the diagonal representation of potentials for the low-frequency FMM for the Helmholtz equation, and for the Yukawa equations (both in two and three dimensions). The scheme we have constructed are an extension of the techniques we constructed in [10] and used in [4], [1] to design an FMM for the Laplace equation in two and three dimensions, respectively; we believe that the resulting implementations are more or less optimal. A paper reporting this work is in preparation, and we are we are enclosing a draft [2].

2. In accordance with our usual practice, we have implemented preliminary (non-adaptive) versions of the Yukawa and low-frequency Helmholtz equations. The resulting algorithms are similar (in terms of speed, accuracy, and complexity) to the algorithm for the Laplace equation reported in [3]. The report describing this work is in preparation.

3. To some extent, during the final stages of this project, we reduced our concentration on the mechanics of the FMM itself, and intensified our investigation of collateral issues, such as discretization of surfaces, connection with the recently developed time-domain solvers, the possibility of direct solvers in the frequency domain, issues arising in the application of our techniques in the high-frequency regime (the latter being, arguably, outside the scope of this project). The principal reason for this redeployment is the fact that the FMM in its original form is rapidly becoming an accepted tool in several areas, and it has not been the purpose of this project to design algorithms for specific applications.

References

[1] H. CHENG, L. GREENGARD, V. ROKHLIN, *A Fast Adaptive Multipole Algorithm in Three Dimensions*, Yale University Technical Report, YALEU/DCS/RR-1158, 1998.

[2] H. CHENG, V. ROKHLIN, N. YARVIN, *Non-linear Optimization, Quadrature, and Interpolation*, Yale University Technical Report, YALEU/DCS/RR-1169, 1998.

[3] L. GREENGARD, V. ROKHLIN, *A new version of the Fast Multipole Method for the Laplace Equation in three dimensions*, Acta Numerica, 1997, pp. 229-269.

[4] T. HRYCAK, V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields*, Research Report 1089, Yale Computer Science Department, 1995.

[5] R. JAKOB-CHIEN, B. ALPERT, *A Fast Spherical Filter with Uniform Resolution*, Journal of Computational Physics, Vol. 136, No. 2, 1997.

[6] J. MA, V. ROKHLIN, S. WANDZURA, *Generalized Gaussian Quadratures For Systems of Arbitrary Functions*, SIAM Journal of Numerical Analysis, v. 33, No. 3, pp. 971-996, 1996.

[7] A. A. MARKOV, *On the limiting values of integrals in connection with interpolation*, Zap. Imp. Akad. Nauk. Fiz.-Mat. Otd. (8) 6 (1898), no. 5 (Russian), pp. 146-230 of [8].

[8] A. A. MARKOV, *Selected papers on continued fractions and the theory of functions deviating least from zero*, OGIZ, Moscow-Leningrad, 1948 (Russian).

[9] Accelerating Fast Multipole Methods For Low Frequency Scattering L. GREENGARD, J. HUANG, V. ROKHLIN, S. WANDZURA, *IEEE Computational Science and Engineering*, v. 5, No. 3, July-September 1998.

[10] N. YARVIN AND V. ROKHLIN, *Generalized Gaussian Quadratures and Singular Value Decompositions of Integral Operators*, Yale University Technical Report YALEU/DCS/ RR-1109 (1996), to appear in *SIAM Journal on Scientific Computing.*

[11] N. YARVIN AND V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields on One-Dimensional Structures*, Yale University Technical Report YALEU/DCS/ RR-1119 (1997), to appear in *SIAM Journal on Numerical Analysis.*

[12] N. YARVIN AND V. ROKHLIN, *A Generalized One-Dimensional Fast Multipole Method with Application to Filtering of Spherical Harmonics*, Yale University Technical Report, YALEU/DCS/RR-1142, 1998, to appear in *Journal of Computational Physics.*

We present a non-linear optimization procedure for the design of Generalized Gaussian Quadratures for a fairly broad class of functions. While some of the components of the algorithm have been published previously, we introduce an improved procedure for the determination of an acceptable initial point for the continuation scheme that stabilizes the Newton-type process used to find the quadratures. The resulting procedure never failed when applied to Chebyshev systems (for which the existence and uniqueness of Generalized Gaussian Quadratures are well-known); it also worked for many non-Chebyshev systems, for which the Generalized Gaussian Quadratures (generally speaking) do not exist. The performance of the algorithm is illustrated with several numerical examples; some of the presented quadratures integrate efficiently large classes of singular functions.

Non-linear Optimization, Quadrature, and Interpolation

H. Cheng, V. Rokhlin, N. Yarvin
Research Report YALEU/DCS/RR-1169
December 17, 1998

The authors were supported in part by DARPA/AFOSR under Contract F49620/97/1/0011, in part by ONR under Grant N00014-96-1-0188, in part by the AFOSR under Contract F49620-97-C-0052, and in part by the AFOSR under Contract F49620-95-C-0075.
Approved for public release: distribution is unlimited.
Keywords: *Non-linear Optimization, Quadratures, Singular Integrands, Interpolation.*

1 Introduction

Quadrature formulae are one of the most developed areas of computational mathematics. They are used both as a stand-alone numerical tool for the evaluation of integrals, and as an analytical apparatus for the design of interpolation schemes, finite element schemes, etc. Most of the quadrature formulae (at least for functions on $\mathbb{R}^1$) currently in use can be separated into three groups:

1. Gaussian quadratures are the optimal tool for the evaluation of integrals of the form

$$\int_a^b \omega(t) \cdot P(t)dt, \tag{1}$$

where P is a polynomial of t (or a function well-approximated by a polynomial), and ω is a (more or less) arbitrary non-negative function $[a, b] \to \mathbb{R}$. Gaussian quadratures are extremely efficient, mathematically elegant, and easy to obtain (see, for example [3]); whenever applicable, they tend to be the numerical tool of choice.

2. Interpolatory quadrature formulae (Newton-Cotes, etc.) are based on approximating the integrand by some standard function (usually, a polynomial), and integrating the latter. These schemes have the advantage that they (usually) do not prescribe the locations of the nodes; they tend to become numerically unstable for high orders.

3. Miscellaneous special-purpose quadratures ("product integration rules", non-standard Richardson extrapolation, etc.) are normally used when the situation precludes the use of more straightforward techniques.

There appears to exist a class of situations where classical approaches fail to produce rapidly convergent schemes. Specifically, suppose that we wish to integrate functions of the form

$$\sum_{k=0}^{n} \phi_k(x) \cdot s_k(x), \tag{2}$$

where ϕ_k are smooth functions (or polynomials) mapping $[0, 1] \to \mathbb{R}$, and the functions $s_k : [0, 1] \to \mathbb{R}$ are known apriori, and have singularities at $x = 0$. In many situations of interest, the functions s_k have *different* singularities at $x = 0$, and the functions ϕ_k *are not known at all*; it is only known that the integrand has the form (2), and its values at points on the interval $[0, 1]$ can be evaluated. While efficient quadratures for functions of the form (2) would have obvious applications in the solution of integral equations, in numerical complex analysis, and in several other areas, the authors have failed to find such an apparatus in the literature.

It has been known for about 100 years that Gaussian quadratures admit a drastic generalization, replacing polynomials with fairly general systems of functions (see [11, 12], [2, 8], [6, 7]). The constructions found in (see [11, 12], [2, 8], [6, 7]) do not easily yield numerical algorithms for the design of such quadrature formulae; algorithms of this type were designed (in some cases) in [10, 15], where the resulting quadrature rules are referred to as Generalized Gaussian Quadratures. The approach is based on the observation that the nodes and weights of Gaussian quadratures satisfy systems of non-linear equations, that these equations

have unique solutions, and that when polynomials are replaced with other systems of functions, similar systems of equations are easily constructed. While for functions of the form (2) the resulting equations are non-linear, overdetermined, and non-unique, in the least squares sense they have unique solutions under surprisingly general conditions (see [10, 15]); Newton-type methods converge in this environment, provided a good initial approximation can be found.

As often happens, in the absence of a good initial approximation, the Newton process fails to converge. To some extent, this problem is remedied by the use of continuation techniques, which turn out to be almost always available when designing quadratures for integrands (2). However, yet another problem is frequently encountered: even though *mathematically* the solution of the non-linear problem is unique for all values of the continuation parameter, *numerically* it is not unique at all. Once the (numerical) rank of the Jacobian of an intermediate problem is sufficiently low, the continuation process breaks down; attempts to use globalized search techniques have not been successful.

The final step in the design of a robust scheme for the construction of Generalized Gaussian Quadratures is described in Section 3.3. It finds an initial approximation for which the Jacobian of the system being solved has an acceptably low condition number. While the reasoning behind this step is partly Heuristic, in our experience it works remarkably well. It never failed for a Chebyshev system (see Section 2.1 below); furthermore, it worked for most of the non-Chebyshev systems we tried it on. For a more detailed discussion of our numerical experience, see Section 5 below, where we also present quadratures for functions with *almost* general power singularities at one end (or both ends) of the interval of integration, and with several other types of singularities.

The paper is structured as follows. Section 2 contains mathematical and numerical preliminaries. In Section 3, we build the numerical apparatus to be used in Section 4 to construct the procedure for the determination of nodes and weights of Generalized Gaussian Quadratures. Section 5 contains several examples of quadratures we have obtained. Finally, in Section 6 we outline several possible extensions of this work.

2 Mathematical and Numerical Preliminaries

2.1 Chebyshev systems

Definition 2.1 *A sequence of functions $\phi_1, \ldots, \phi_n$ will be referred to as a* Chebyshev system *on the interval $[a, b]$ if each of them is continuous and the determinant*

$$\begin{vmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_n(x_1) & \cdots & \phi_n(x_n) \end{vmatrix} \tag{3}$$

is nonzero for any sequence of points $x_1, \ldots, x_n$ such that $a \leq x_1 < x_2 \ldots < x_n \leq b$.

An alternate definition of a Chebyshev system is that any linear combination of the functions with nonzero coefficients must have no more than n zeros.

A related definition is that of an extended Chebyshev system.

Definition 2.2 *Given a set of functions $\phi_1, \ldots, \phi_n$ which are continuously differentiable on an interval $[a, b]$, and given a sequence of points $x_1, \ldots, x_n$ such that $a \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq b$, let the sequence $m_1, \ldots, m_n$ be defined by the formulae*

$$\begin{aligned} m_1 &= 0, \\ m_j &= 0 && \text{if } j > 1 \text{ and } x_j \neq x_{j-1}, \\ m_j &= j - 1 && \text{if } j > 1 \text{ and } x_j = x_{j-1} = \ldots = x_1, \\ m_j &= k && \text{if } j > k + 1 \text{ and } x_j = x_{j-1} = \ldots = x_{j-k} \neq x_{j-k-1}. \end{aligned} \tag{4}$$

Let the matrix $C(x_1, \ldots, x_n) = [c_{ij}]$ be defined by the formula

$$c_{ij} = \frac{d^{m_j} \phi_i}{dx^{m_j}}(x_j), \tag{5}$$

in which $\frac{d^0 \phi_i}{dx^0}(x_j)$ is taken to be the function value $\phi_i(x_j)$. Then $\phi_1, \ldots, \phi_n$ will be referred to as an extended Chebyshev system *on $[a, b]$ if the determinant $|C(x_1, \ldots, x_n)|$ is nonzero for all such sequences x_i.*

Remark 2.1 *It is obvious from Definition 2.2 that an extended Chebyshev system is a special case of the Chebyshev system. The additional constraint is that the successive points x_i at which the function is sampled to form the matrix may be identical; in that case, for each duplicated point, the first corresponding column contains the function values, the second column contains the first derivatives of the functions, the third column contains the second derivatives of the functions, and so forth; this matrix must also be nonsingular.*

Examples of Chebyshev and extended Chebyshev systems include the following (additional examples can be found in [7]).

Example 2.1 *The powers $1, x, x^2, \ldots, x^n$ form an extended Chebyshev system on the interval $(-\infty, \infty)$.*

Example 2.2 *The exponentials $e^{-\lambda_1 x}, e^{-\lambda_2 x}, \ldots, e^{-\lambda_n x}$ form an extended Chebyshev system for any $\lambda_1, \ldots, \lambda_n > 0$ on the interval $[0, \infty)$.*

Example 2.3 *The functions $1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx$ form a Chebyshev system on the interval $[0, 2\pi)$.*

2.2 Generalized Gaussian quadratures

The quadrature rules considered in this paper are expressions of the form

$$\sum_{j=1}^{n} w_j \cdot \phi(x_j) \tag{6}$$

where the points $x_j \in \mathbb{R}$ and coefficients $w_j \in \mathbb{R}$ are referred to as the nodes and weights of the quadrature, respectively. They serve as approximations to integrals of the form

$$\int_a^b \phi(x) \; cdot\omega(x) dx \tag{7}$$

where ω has the form

$$\omega(x) = \tilde{\omega}(x) + \sum_{j=1}^{m} \mu_j \cdot \delta(x - \chi_j), \tag{8}$$

with m a non-negative integer, $\tilde{\omega} : [a, b] \to \mathbb{R}$ an integrable non-negative function, $\chi_1, \chi_2, \ldots, \chi_m$ points on the interval $[a, b]$, $\mu_1, \mu_2, \ldots, \mu_m$ positive real coefficients, and δ the Dirac δ-function on $\mathbb{R}$.

Remark 2.2 *Obviously, (8) defines ω to be a linear combination of a non-negative function with a finite collection of δ-functions with positive coefficients. In a mild abuse of terminology, throughout this paper, we will be referring to ω as a non-negative function.*

Quadratures are typically chosen so that the quadrature (6) is equal to the desired integral (7) for some set of functions, commonly polynomials of some fixed order. Of these, the classical Gaussian quadrature rules consist of n nodes and integrate polynomials of order $2n-1$ exactly. In [10], the notion of a Gaussian quadrature was generalized as follows:

Definition 2.3 *A quadrature formula will be referred to as* Gaussian *with respect to a set of $2n$ functions $\phi_1, \ldots, \phi_{2n} : [a, b] \to \mathbb{R}$ and a weight function $\omega : [a, b] \to \mathbb{R}^+$, if it consists of n weights and nodes, and integrates the functions ϕ_i exactly with the weight function ω for all $i = 1, \ldots, 2n$. The weights and nodes of a Gaussian quadrature will be referred to as Gaussian weights and nodes respectively.*

The following theorem appears to be due to Markov [11, 12]; proofs of it can also be found in [8] and [7] (in a somewhat different form).

Theorem 2.1 *Suppose that the functions $\phi_1, \ldots, \phi_{2n} : [a, b] \to \mathbb{R}$ form a Chebyshev system on $[a, b]$. Suppose in addition that $\omega : [a, b] \to \mathbb{R}$ is defined by (8), and that either*

$$\int_a^b \tilde{\omega}(x)dx > 0, \tag{9}$$

or $m \geq n$ (or both). Then there exists a unique Gaussian quadrature for $\phi_1, \ldots, \phi_{2n}$ on $[a, b]$ with respect to the weight function ω. The weights of this quadrature are positive.

2.3 Quadrature and Interpolation

As is well-known, when Gaussian nodes on the interval $[-1, 1]$ are used for interpolation (for example, via the Lagrange formula), the resulting procedure is numerically stable. Furthermore, the precision obtained via Gaussian (Lagrange) interpolation is almost as high as that obtained via Chebyshev interpolation (see, for example, [4]). Generally, given a weight function ω, the nodes of Gaussian quadratures corresponding to ω lead to interpolation formulae that are stable in an appropriately chosen norm. In this subsection, we formalize this fact for both Gaussian and many Generalized Gaussian quadratures. The analytical tool of this subsection is the following obvious theorem.

Theorem 2.2 *Suppose that the function $\omega : [a,b] \to \mathbb{R}$ is non-negative, and the functions $\phi_1, \phi_2, \ldots, \phi_n : [a,b] \to \mathbb{R}$ are orthonormal with respect to the weight function ω, i.e.*

$$\int_a^b \omega(x) \cdot \phi_j(x) \cdot \phi_i(x) dx = \delta_{ij} \tag{10}$$

for all $i, j = 1, 2, \ldots, n$ (δ_{ij} denotes Kroneker's δ-function). Suppose further that the n-point quadrature rule $x_1, x_2, \ldots, x_n$, $w_1, w_2, \ldots, w_n$, is such that $w_i > 0$ for all $1 \leq i \leq n$. Finally, suppose that

$$\sum_{k=1}^{n} w_k \cdot \phi_i(x_k) \cdot \phi_j(x_k) = \delta_{ij} \tag{11}$$

for all $i, j = 1, 2, \ldots, n$. Then the $n \times n$-matrix A defined by the formula

$$A_{ij} = \sqrt{w_j} \cdot \phi_i(x_j), \tag{12}$$

is orthogonal.

Suppose now that we would like to construct an interpolation formula on the interval $[a,b]$ for functions of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i \cdot \phi_i(x), \tag{13}$$

with $\alpha_1, \alpha_2, \ldots, \alpha_n$ arbitrary real coefficients. In other words, suppose that we are given the values $f_1, f_2, \ldots, f_n$ of a function f at a collection of points $x_1, x_2, \ldots, x_n$, and that it is known that f is defined by the formula (13), but the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ are not known; we would like to be able to evaluate f at arbitrary points on $[a,b]$. The obvious way to do so is to observe that the values $f_1, f_2, \ldots, f_n$ are linear functions of the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ (due to (13)); evaluating (13) at the points $x_1, x_2, \ldots, x_n$, we obtain the system of equations

$$f_j = \sum_{i=1}^{n} \alpha_i \cdot \phi_i(x_j), \tag{14}$$

with $j = 1, 2, \ldots, n$. Defining the $n \times n$-matrix B by the formula

$$b_{j,i} = \phi_i(x_j), \tag{15}$$

we rewrite (14) in the form

$$F = B\alpha, \tag{16}$$

with the vectors $\alpha, F \in \mathbb{R}^n$ defined by the formulae

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n), \tag{17}$$

$$F = (f_1, f_2, \ldots, f_n). \tag{18}$$

Now, as long as the matrix B is non-singular, we can evaluate the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ via the formula

$$\alpha = B^{-1}F, \tag{19}$$

and use (13) to evaluate f at arbitrary points on $[a,b]$. Of course, in actual numerical calculations, it is not sufficient for B to be invertible; its condition number must not be too high. The following observation is the principal purpose of this subsection.

Observation 2.3 *Under the conditions of Theorem 2.2,*

$$A = D \circ B, \tag{20}$$

with D the diagonal matrix defined by the formula

$$D_{i,i} = \sqrt{w_i}, \tag{21}$$

and

$$\alpha = A^* DF \tag{22}$$

(due to the combination of (19) with (20)). In other words, given the table of values $f_1, f_2, \ldots, f_n$ of the function f at the nodes $x_1, x_2, \ldots, x_n$, one obtains the coefficients of the expansion (13) by applying to the vector F the product of two matrices; the first of these matrices is orthogonal, and the second is diagonal; the diagonal elements of the latter are square roots of (positive) weights of the n-point quadrature formula exact for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$.

Remark 2.4 *While at first glance the above observation appears to be very limited in its scope (since it relies on the quadrature formula being exact for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$), in reality it means that whenever the nodes of a Generalized Gaussian quadrature formula are used as interpolation nodes, the resulting interpolation formula tends to be stable. The reason for this happy coincidence is the fact that the matrix A (see (12) above)) need not be orthogonal for the stability of the interpolation formula; it only needs to be well-conditioned. Thus, as long as the quadrature formula is reasonably accurate for all pairwise products of the functions $\phi_1, \phi_2, \ldots, \phi_n$, the matrix A is close to being orthogonal; therefore, the condition number of A is close to unity, and the interpolation based on the nodes $x_1, x_2, \ldots, x_n$ is stable.*

2.4 Convergence of Newton's method

In this section, we observe that the nodes and the weights of a Gaussian quadrature satisfy a simple system of nonlinear equations. We then prove that the Newton method for this system of equations is always quadratically convergent, provided the functions to be integrated constitute an extended Chebyshev system.

Given a set of functions $\phi_1, \ldots, \phi_{2n}$ and a weight function ω, the Gaussian quadrature is defined by the system of equations

$$\begin{aligned}
\sum_{j=1}^{n} w_j \cdot \phi_1(x_j) &= \int_a^b \phi_1(x) \cdot \omega(x) dx, \\
\sum_{j=1}^{n} w_j \cdot \phi_2(x_j) &= \int_a^b \phi_2(x) \cdot \omega(x) dx, \\
&\vdots \\
\sum_{j=1}^{n} w_j \cdot \phi_{2n}(x_j) &= \int_a^b \phi_{2n}(x) \cdot \omega(x) dx,
\end{aligned} \tag{23}$$

(see Definition 2.3). Let the left hand sides of these equations be denoted by f_1 through f_{2n}. Then each f_i is a function of the weights $w_1, \ldots, w_n$ and nodes $x_1, \ldots, x_n$ of the quadrature. Its partial derivatives are given by the obvious formulae

$$\frac{\partial f_k}{\partial w_i} = \phi_k(x_i), \tag{24}$$

$$\frac{\partial f_k}{\partial x_i} = w_i \cdot \phi_k'(x_i). \tag{25}$$

Thus, the Jacobian matrix of the system (23) is

$$J(x_1, \ldots, x_n, w_1, \ldots, w_n) = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) & w_1\phi_1'(x_1) & \cdots & w_n\phi_1'(x_n) \\ \vdots & & \vdots & \vdots & & \vdots \\ \phi_{2n}(x_1) & \cdots & \phi_{2n}(x_n) & w_1\phi_{2n}'(x_1) & \cdots & w_n\phi_{2n}'(x_n) \end{pmatrix}. \tag{26}$$

Lemma 2.3 *Suppose that the functions $\phi_1, \ldots, \phi_{2n}$ form an extended Chebyshev system. Let the Gaussian quadrature for these functions be denoted by $\hat{w}_i$ and $\hat{x}_i$. Then the determinant of J is nonzero at the point which constitutes the Gaussian quadrature; in other words, $|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| \neq 0$.*

Proof. It is immediately obvious from (26) that

$$|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| = \hat{w}_1 \cdot \hat{w}_2 \cdot \cdots \cdot \hat{w}_{n-1} \cdot \hat{w}_n \cdot \begin{vmatrix} \phi_1(\hat{x}_1) & \cdots & \phi_1(\hat{x}_n) & \phi_1'(\hat{x}_1) & \cdots & \phi_1'(\hat{x}_n) \\ \vdots & & \vdots & \vdots & & \vdots \\ \phi_{2n}(\hat{x}_1) & \cdots & \phi_{2n}(\hat{x}_n) & \phi_{2n}'(\hat{x}_1) & \cdots & \phi_{2n}'(\hat{x}_n) \end{vmatrix}. \tag{27}$$

If $\phi_1, \ldots, \phi_{2n}$ form an extended Chebyshev system, then by Theorem 2.1, the weights $\hat{w}_1, \ldots, \hat{w}_n$ of the Gaussian quadrature are positive. In addition, by the definition of an extended Chebyshev system, the determinant in the right hand side of (27) is nonzero. Thus

$$|J(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_n)| \neq 0. \tag{28}$$

□

Using the inverse function theorem, we immediately obtain the following corollary:

Corollary 2.4 *Under the conditions of Lemma 2.3, the Gaussian weights and nodes depend continuously on the weight function.*

2.5 Singular value decomposition

The singular value decomposition (SVD) is a ubiquitous tool in numerical analysis, given for the case of real matrices by the following lemma (see, for instance, [13] for more details).

Lemma 2.5 *For any $n \times m$ real matrix A, there exist, for some integer p, an $n \times p$ real matrix U with orthonormal columns, an $m \times p$ real matrix V with orthonormal columns, and a $p \times p$ real diagonal matrix $S = [s_{ij}]$ whose diagonal entries are non-negative, such that $A = U \cdot S \cdot V^*$ and that $s_{ii} \geq s_{i+1,i+1}$ for all $i = 1, \ldots, p-1$.*

The diagonal entries s_{ii} of S are called singular values; the columns of the matrix V are called right singular vectors; the columns of the matrix U are called left singular vectors.

2.6 Singular value decomposition of a sequence of functions

A similar decomposition exists (see [5, 16]) if the columns of the matrix A are replaced by functions:

Theorem 2.6 *Suppose that the functions $\phi_1, \phi_2, \ldots, \phi_n : [a,b] \to \mathbb{R}$ are square integrable. Then there exist a finite orthonormal sequence of functions $u_1, u_2, \ldots, u_p : [a,b] \to \mathbb{R}$, an $n \times p$ matrix $V = [v_{ij}]$ with orthonormal columns, and a sequence $s_1 \geq s_2 \geq \ldots \geq s_p > 0 \in \mathbb{R}$, for some integer p, such that*

$$\phi_j(x) = \sum_{i=1}^{p} u_i(x) s_i v_{ij}, \tag{29}$$

for all $x \in [a,b]$ and all $j = 1, \ldots, n$. The sequence $\{s_i\}$ is uniquely determined by K.

By analogy to the finite-dimensional case, we refer to this factorization as the singular value decomposition. We refer to the functions $\{u_i\}$ as singular functions, to the columns of the matrix V as singular vectors, and to the numbers $\{s_i\}$ as singular values.

A popular application of the Singular Value Decomposition is for the purpose of "compressing" data. Specifically, it often happens that while the total number n of functions is large, almost all of the coefficients s_j in the decomposition (29) are negligibly small. In such cases, (29) is truncated after a small number (say, p_0) of terms, and the resulting expansion

$$\phi_j(x) = \sum_{i=1}^{p_0} u_i(x) \cdot s_i \cdot v_{ij} \tag{30}$$

is viewed as a compact representation of the original family of functions $\phi_1, \phi_2, \ldots, \phi_n$.

The following theorem states that given a sequence of functions on the interval $[a,b]$, their decomposition of the form (30), and a quadrature formula with positive weights on the interval $[a,b]$, the accuracy of the quadrature for the functions $\phi_1, \phi_2, \ldots, \phi_n$ is determined by its accuracy for the singular functions u_j, corresponding to non-trivial singular values. Its proof is an exercise in elementary linear algebra, and is omitted.

Theorem 2.7 *Suppose that under the conditions of Theorem 2.6, ϵ is a positive real number, $1 < p_0 < n$ is an integer, and*

$$\sum_{i=p_0+1}^{n} s_i^2 < \frac{\epsilon^2}{4}. \tag{31}$$

Suppose further that the m-point quadrature formula $\{x_i, w_i\}$ integrates the functions u_i exactly, i.e.

$$\sum_{j=1}^{m} w_j \cdot u_i(x_j) = \int_a^b u_i(x)\, dx \tag{32}$$

for all $i = 1, 2, \ldots, p_0$, and that all of the weights $w_1, \ldots, w_m$ are positive. Then for each $i = 1, 2, \ldots, n$,

$$\left| \sum_{j=1}^{m} w_j \cdot \phi_i(x_j) - \int_a^b \phi_i(x)\, dx \right| < \epsilon \cdot ||\phi_i||_{L^2}. \tag{33}$$

3 Numerical Apparatus

3.1 Continuation method

In order for Newton's method to converge, the starting point provided to it must be close to the desired solution. One scheme for generating such starting points is the continuation method, described below.

Suppose that in addition to the function $F : \mathbb{R}^n \to \mathbb{R}^n$ whose zero is to be found, another function $G : [0,1] \times \mathbb{R}^n \to \mathbb{R}^n$ is available which possesses the following properties:

- 1. For any $x \in \mathbb{R}^n$,
$$G(1,x) = F(x). \tag{34}$$
- 2. The solution of the equation
$$G(0,x) = 0 \tag{35}$$
is known.
- 3. For all $t \in [0,1]$, the equation
$$G(t,x) = 0 \tag{36}$$
has a unique solution x at which the conditions for Newton's method to converge are satisfied.
- 4. The solution x is a continuous function of t.

If these conditions are met, an algorithm for the solution of the equation

$$F(x) = 0 \tag{37}$$

is as follows. Let the points t_i, for $i = 1, \ldots, m$, be defined by the formula $t_i = i/m$. Solve in succession the equations

$$\begin{aligned} G(t_1,x) &= 0, \\ G(t_2,x) &= 0, \\ &\vdots \\ G(t_m,x) &= 0 \end{aligned} \tag{38}$$

using Newton's method, with the starting point for Newton's method for each equation taken to be the solution of the preceding equation. Due to (34), the solution x of the final equation $G(t_m,x) = 0$ is identical to the solution of (37); obviously, for sufficiently large m, Newton's method is guaranteed to converge at each step.

Remark 3.1 *In practice, it is desirable to choose the smallest m for which the above algorithm will work, in order to reduce the computational cost of the scheme. On the other hand, the largest step $(t_i - t_{i-1})$ for which the Newton method will converge commonly varies as a function of t. Thus the algorithm described in this paper uses an adaptive version of the scheme.*

3.2 Continuation scheme

The continuation scheme used is as follows. Let the weight functions $\omega : [0,1] \times [a,b] \to \mathbb{R}^+$ be defined by the formula

$$\omega(\alpha, x) = \alpha\omega_1(x) + (1-\alpha)\sum_{j=1}^{n}\delta(x - c_j), \tag{39}$$

where ω_1 is the weight function for which a Gaussian quadrature is desired, δ denotes the Dirac delta function, and the points $c_j \in [a,b]$ are arbitrary distinct points. These weight functions have the following properties:

- 1. With $\alpha = 1$, the weight function is equal to the desired weight function ω_1, due to (39).
- 2. With $\alpha = 0$, the Gaussian weights and nodes are

$$w_j = 1, \tag{40}$$
$$x_j = c_j, \tag{41}$$

  for $j = 1, \ldots, n$, whatever the functions ϕ_i are (since $\omega(0,x) = 0$, unless $x = c_j$ for some $j \in [1,n]$).
- 3. The quadrature weights and nodes depend continuously on α (by Corollary 2.4).

The intermediate problems which the continuation method solves are the Gaussian quadratures relative to the weight functions $\omega(\alpha, *)$. The scheme starts by setting $\alpha = 0$, then increases α in an adaptive manner until $\alpha = 1$, as follows. A current step size is maintained, by which α is incremented after each successful termination of Newton's method. After each unsuccessful termination of Newton's method, the step size is halved and the algorithm restarts from the point yielded by the last successful termination. After a certain number of successful steps, the current step size is doubled. (Experimentally, the current problem was found to be well suited to an aggressive mode of adaption: in the authors' implementation, the initial value of the step size was chosen to be 0.5, and the step size was doubled after two successful terminations of Newton's method.)

3.3 Starting points

The choice of the points c_j was left indefinite above. In exact arithmetic, and applied to a Chebyshev system, the algorithm would converge for any choice of distinct points (see Lemma 2.3). However, the number of steps of the continuation method, and thus the speed of execution, is affected by the choice. More importantly, the numerical stability of the scheme might be compromised due to poor conditioning of the matrix J (see (26)). Indeed, while Lemma 2.3 guarantees that the matrix J is non-singular, it says nothing about its condition number. In addition, we will be applying the algorithm to cases where the conditions of Lemma 2.3 are not satisfied. For these reasons, the following method of choosing the starting points was adopted. The method seeks to create a matrix J that is well-conditioned. It is a pivoted Gram-Schmidt orthogonalization, altered to operate on pairs of vectors:

- 1. Choose a set of points $x_1, x_2, \ldots, x_m$ on the interval of integration $[a, b]$, such that each of the functions $\phi_1, \phi_2, \ldots, \phi_n$, and each of their derivatives, can be interpolated on $[a, b]$ in a well-conditioned manner from values at these points.
- 2. Create a matrix $\tilde{J}$, of the same form as (26), where the points $\{x_j\}$ which determine the columns are the points chosen in step 1. (This matrix thus has $2m$ columns.)
- 3. Perform the following sequence of operations n times:
  - a) choose the point x_j for which the two columns corresponding to x_j have the largest size. (The issue of what "size"to use is discussed below.)
  - b) orthogonalize the remaining columns to both of those two columns.

The points x_j chosen in step (3a) are then the starting points c_j used in the continuation method.

The algorithm as specified above is for exact arithmetic. As with Gram-Schmidt, the algorithm is numerically unstable, but can be stabilized by an additional re-orthogonalization: after step (3a), re-orthogonalize the two new pivot columns to all of the previously chosen pivot columns.

Remark 3.2 *The "size of two columns" that was used for step (3a) is the sum of the norms of the columns, after the second column has been orthogonalized to the first. This poses the obvious danger that one of the two columns chosen might have a small norm, which was covered up by a large norm of its companion. This would render it unsuitable for pivoting; this danger was never realized in our numerical experiments, but if it were, the obvious remedy would be to attempt to change the definition of the "size". The authors have not investigated this issue in detail.*

3.4 Nested Legendre discretizations of finite sequences of functions

In this paper, we will be confronted with finite sequences of functions $\phi_1, \phi_2, \ldots \phi_n$ on the interval $[a, b]$, possessing the following properties:

- 1. The total number n of functions ϕ_i is reasonably large (e.g. 10000).
- 2. The rank of the set $\phi_1, \phi_2, \ldots \phi_n$, is low (e.g. 40), to high precision.
- 3. Each of the functions $\phi_1, \phi_2, \ldots \phi_n$ is analytic on the interval $[a, b]$, except at a finite (small) number of points; $\phi_i \in L^1[a, b]$ for all $i = 1, 2, \ldots, n$.

Now, if we wish to handle (interpolate, integrate, differentiate, etc.) numerically functions of the form

$$\psi(x) = \sum_{i=1}^{n} \alpha_i \cdot \phi_i, \tag{42}$$

often it is not convenient to represent them by collections of coefficients $\alpha_1, \alpha_2, \ldots \alpha_n$. Indeed, if the functions $\phi_1, \phi_2, \ldots \phi_n$ are linearly dependent, the number of coefficients α_i necessary to

represent them in the form (42) might be grossly excessive, compared to the actual complexity of the function to be represented. Furthermore, the coefficients α_i by themselves provide no mechanism for the integration, interpolation, etc. of functions of the form (42); each time such procedures have to be performed, one has to recompute the original functions $\phi_1, \phi_2, \ldots \phi_n$. Since the latter is often expensive or impossible, it is desirable to have a purely numerical procedure for representing sums of the form (42). Preferably, the scheme should use no information about the functions ϕ_i, except for their values at a finite (preferably, not very large) collection of points on $[a, b]$.

When the functions ϕ_i are smooth, a widely used tool for representing them is Chebyshev interpolation: a sufficiently large integer m is chosen, the functions $\phi_1, \phi_2, \ldots \phi_n$ are tabulated at m Chebyshev nodes on $[a, b]$, and obtained at all other points on $[a, b]$ via standard interpolation procedures. While Chebyshev nodes are an extremely good choice, they are not the only one; for example, Gaussian (Legendre) nodes are almost as efficient as the Chebyshev ones when the functions are to be interpolated, and twice as efficient when the functions are to be integrated (see, for example, [4]). When the behavior of the functions ϕ_i is very non-uniform over the interval $[a, b]$, Chebyshev (Gaussian, etc.) interpolation becomes inefficient; for singular functions it is liable to fail completely. In such cases, adaptive Chebyshev interpolation is used, whereby the interval is subdivided into a collection of subintervals, so that on each subinterval, all of the functions ϕ_i are accurately approximated by a Chebyshev expansion of low order; needless to say, most of the time, such subdivisions are performed automatically. When some (or all) of the functions ϕ_i have singularities on the interval $[a, b]$, schemes of this type cluster the subintervals near each singularity, until the subinterval nearest to the singularity is so small as to be ignorable for the purposes of the calculations to be performed.

In the first stage of the algorithm we use, we build a nested Chebyshev discretization of the interval [a,b] for each of the functions ϕ_i. In the second stage, all such discretizations are merged to obtain a single discretization by which all of the functions ϕ_i are adequately represented. In the third stage, n *Legendre* nodes are constructed on each of the obtained intervals.

Stage 1

- 1. Choose the precision ϵ and some reasonably large m (in actual computations, we use $m = 16$).
- 2. Construct the m Chebyshev nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, on the interval $[a, b]$. Evaluate ϕ at the nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, obtaining the values $\phi_1^{[a,b]}, \phi_2^{[a,b]}, \ldots, \phi_m^{[a,b]}$.
- 3. Subdivide the interval $[a, b]$ into the subintervals $[a, (a+b)/2]$, $[(a+b)/2, b]$. Construct the Chebyshev nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}$ on the interval $[a, (a+b)/2]$, and the Chebyshev nodes $x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}, \ldots, x_m^{[(a+b)/2,b]}$ on the interval $[(a+b)/2, b]$. Evaluate the function ϕ at the nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}, x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}, \ldots, x_m^{[(a+b)/2,b]}$, obtaining the values $\phi_1^{[a,(a+b)/2]}, \phi_2^{[a,(a+b)/2]}, \ldots, \phi_m^{[a,(a+b)/2]}, \phi_1^{[(a+b)/2,b]}, \phi_2^{[(a+b)/2,b]}, \ldots, \phi_m^{[(a+b)/2,b]}$, respectively.
- 4. Interpolate the values of the function ϕ from the nodes $x_1^{[a,b]}, x_2^{[a,b]}, \ldots, x_m^{[a,b]}$, on the interval $[a, b]$ to the nodes $x_1^{[a,(a+b)/2]}, x_2^{[a,(a+b)/2]}, \ldots, x_m^{[a,(a+b)/2]}, x_1^{[(a+b)/2,b]}, x_2^{[(a+b)/2,b]}$,

$\ldots$, $x_m^{[(a+b)/2,b]}$ on the intervals $[a,(a+b)/2]$, $[(a+b)/2,b]$. If the interpolated values agree to the precision ϵ with the values $\phi_1^{[a,(a+b)/2]}$, $\phi_2^{[a,(a+b)/2]}$, $\ldots$, $\phi_m^{[a,(a+b)/2]}$, $\phi_1^{[(a+b)/2,b]}$, $\phi_2^{[(a+b)/2,b]}$, $\ldots$, $\phi_m^{[(a+b)/2,b]}$ calculated directly in Step 2 above, the algorithm concludes that the function ϕ is adequately resolved by the m Chebyshev nodes on the interval $[a,b]$; otherwise, the procedure is repeated recursively for each of the subintervals $[a,(a+b)/2]$, $[(a+b)/2,b]$.

Stage 2

- 1. Store the ends (left and right) of all subintervals in all subdivisions in a single array a. Sort the elements of a; remove multiple elements in a. The resulting array of points on the interval $[a,b]$ (including the points a,b) is the array of ends of subintervals of the final subdivision.

Stage 3

- 1. Construct an m-point Legendre discretization of each of the subintervals obtained in Stage 2 above.

Remark 3.3 *In the algorithm above, we use Chebyshev discretizations in Stage 1 to construct the subdivision of the interval $[a,b]$; in subsequent calculations we use Legendre discretizations. The reason for this choice is that the interpolations in Stage 1 are carried out more efficiently with Chebyshev discretizations, via the Discrete Cosine Transform and related tools; the Legendre discretizations used subsequently lead to linear interpolation schemes that preserve inner products (see following subsection).*

Remark 3.4 *The scheme of this subsection is a fairly reliable apparatus for the automatic discretization of sets of (more or less) arbitrary user-specified functions. While it is very easy to construct counterexamples in which the algorithm will fail to resolve some (or all) of the input functions, this problem has never been encountered in our practice.*

3.5 Approximation of SVD of a sequence of functions

This section describes a numerical procedure for computing an approximation to the singular value decomposition of a sequence of functions. The algorithm uses quadratures possessing the following property.

Definition 3.1 *We will say that the combination of a quadrature and an interpolation scheme* preserves inner products *on an interval $[a,b]$ if it possesses the following properties.*

- *1. The nodes of the quadrature are identical to the nodes of the interpolation scheme.*
- *2. The function which is output by the interpolation scheme depends in a linear fashion on the values input to the interpolation scheme.*

- *3. The quadrature integrates exactly any product of two interpolated functions; that is, for any two functions $f, g : [a, b] \to \mathbb{R}$ produced by the interpolation scheme, the integral*

$$\int_a^b f(x) \cdot g(x) dx \tag{43}$$

is computed exactly by the quadrature.

Quadratures and interpolation schemes possessing this property include:

Example 3.1 *The combination of a (classical) Gaussian quadrature at Legendre nodes and polynomial interpolation at the same nodes preserves inner products, since polynomial interpolation on n nodes produces an interpolating polynomial of order $n-1$, the product of two such polynomials is a polynomial of order $2n-2$, and a Gaussian quadrature integrates exactly all polynomials up to order $2n-1$.*

Example 3.2 *If an interval is broken into several subintervals, and a quadrature and interpolation scheme preserving inner products is used on each subinterval, then the arrangement as a whole preserves inner products on the original interval. (This follows directly from the definition.)*

Example 3.3 *The combination of the trapezoidal rule on the interval $[0, 2\pi]$, and Fourier interpolation (using the interpolation functions $1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots, \cos nx, \sin nx$) preserves inner products.*

The algorithm described below takes as input a sequence of functions $\phi_1, \phi_2, \ldots, \phi_n : [a, b] \to \mathbb{R}$. It uses as a tool a quadrature and a linear interpolation scheme on the interval $[a, b]$ preserving inner products; the weights and nodes of this quadrature will be denoted by $w_1, \ldots, w_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in [a, b]$ respectively. As will be shown below, the accuracy of the algorithm is then determined by the accuracy to which the interpolation scheme approximates the functions $\phi_1, \phi_2, \ldots, \phi_n$.

The output of the algorithm is a sequence of functions $u_1, \ldots, u_p : [a, b] \to \mathbb{R}$, a sequence of vectors $v_1, \ldots, v_p \in \mathbb{R}^n$, and a sequence of singular values $s_1, \ldots, s_p \in \mathbb{R}$, forming an approximation to the singular value decomposition of $\phi_1, \phi_2, \ldots, \phi_n$.

Description of the algorithm:

- 1. Construct the $n \times m$ matrix $A = [a_{ij}]$ defined by the formula

$$a_{ij} = \phi_j(x_i) \cdot \sqrt{w_i}. \tag{44}$$

- 2. Compute the singular value decomposition of A, to produce the factorization

$$A = U \circ S \circ V^*, \tag{45}$$

where $U = [u_{ij}]$ is an $n \times p$ matrix with orthonormal columns, $V = [v_{ij}]$ is an $m \times p$ matrix with orthonormal columns, and S is a $p \times p$ diagonal matrix whose j'th diagonal entry is s_j.

- 3. Construct the $n \times p$ values $u_k(x_i)$ defined by

$$u_k(x_i) = u_{ik}/\sqrt{w_i}. \tag{46}$$

- 4. For any desired point $x \in [a,b]$, evaluate the functions $u_k : [a,b] \to \mathbb{R}$ using the interpolation scheme on $[a,b]$.

The proof of the following theorem can be found (in a considerably more general form) in [15].

Theorem 3.1 *Suppose that the combination of the quadrature and interpolation scheme with weights and nodes $w_1, \ldots, w_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in [a,b]$, respectively, preserves inner products on $[a,b]$. For any function $K : [a,b] \times [c,d] \to \mathbb{R}$, let $u_i : [a,b] \to \mathbb{R}$, $v_{ij} \in \mathbb{R}$, and $s_i \in \mathbb{R}$ be defined in (44)-(46), for all $i = 1, \ldots, p$. Then*

- *1. The functions u_i are orthonormal, i.e.*

$$\int_a^b u_i(x) u_k(x) dx = \delta_{ik} \tag{47}$$

for all $i, k = 1, \ldots, p$, with δ_{ik} the Kronecker symbol.

- *2. The columns of V are orthonormal, i.e.*

$$\sum_{j=1}^{n} v_{ij} v_{kj} dx = \delta_{ik} \tag{48}$$

for all $i, k = 1, \ldots, p$.

- *3. The sequence of functions $\tilde{\phi}_1, \tilde{\phi}_2, \ldots, \tilde{\phi}_n : [a,b] \to \mathbb{R}$ defined by*

$$\tilde{\phi}_k(x) = \sum_{j=1}^{p} s_j u_j(x) v_{jk}, \tag{49}$$

is identical to the sequence of functions produced by sampling the functions $\phi_1, \phi_2, \ldots, \phi_n$ at the points $\{x_i\}$, then interpolating with the interpolation scheme on $[a,b]$.

4 Numerical Algorithm

This section describes a numerical algorithm for the evaluation of nodes and weights of generalized Gaussian quadratures. The algorithm's input are a sequence of functions $\phi_1, \ldots, \phi_{2n} : [a,b] \to \mathbb{R}$, and the precision ϵ to which the quadratures are to be calculated; its output is the weights and nodes of the quadrature. The functions ϕ_i are supplied by the user in the form of a subroutine, with input parameters (x,i), and output parameter $\phi_i(x)$. The algorithm uses the components described the preceding section.

- 1. The interval $[a,b]$ is discretized via the scheme described in Subsection 3.4, so that all functions ϕ_1, $\phi_2, \ldots,$ ϕ_n are represented to the precision ϵ.

- 2. All of the functions ϕ_1, $\phi_2, \ldots$, ϕ_n are tabulated at the nodes of the discretization obtained in p. 1 above, and the Singular Value Decomposition is obtained of the sequence of functions ϕ_1, $\phi_2, \ldots$, ϕ_n via the scheme described in Subsection 3.5; we will be denoting the obtained singular values by $\lambda_1, \lambda_2, \ldots$.

- 3. Denoting by k the positive integer number such that $\lambda_{2 \cdot k+1} \leq \epsilon \leq \lambda_{2 \cdot k-1}$, we observe that any quadrature formulae with positive coefficients that integrates the obtained singular functions u_1, $u_2, \ldots$ $u_{2 \cdot k}$ exactly, will integrate all of the functions ϕ_1, $\phi_2, \ldots$, ϕ_n with precision ϵ (see Theorem 2.7 in Subsection 2.5). The remainder of the algorithm is devoted to constructing a k-point quadrature formula that will integrate the functions u_1, $u_2, \ldots$ $u_{2 \cdot k}$ exactly.

- 4. The scheme of Subsection 3.3 is used to find the starting nodes $x_1^0, x_2^0, \ldots, x_k^0$ for the continuation process of Subsection 3.2.

- 5. An adaptive version of the continuation method of Subsection 3.2 is used to obtain the k-point quadrature for the functions u_1, $u_2, \ldots$, $u_{2 \cdot k}$; on each step, the Newton algorithm described in Subsection 2.4 is used to solve the system (23) defining the nodes and roots of the quadrature formula.

Remark 4.1 *We would like to reiterate that the quadrature formulae produced by the procedure of this section do not integrate the user-specified functions ϕ_1, $\phi_2, \ldots$, ϕ_n exactly; instead, they produce approximations to the integrals. Needless to say, the two are indistinguishable, as long as the chosen precision ϵ is less than the machine precision.*

5 Numerical examples

A variety of quadratures were generated via the algorithm of this paper; several of these are presented below to illustrate its performance. In Examples 5.1, 5.2, the calculations were performed in extended precision (Fortran `REAL*16`) arithmetic, to assure full double precision in the obtained result. In Example 5.3, the calculations were performed in double precision, since the accuracy of the quadrature listed in Table 5 is only 9 digits.

Example 5.1 An obvious problem of interest is the integration on an interval of functions that have a singularity at one end of that interval (or at both ends); of particular interest are power and logarithmic singularities. Many techniques have been proposed for dealing with such problems (see, for example, [1]). While some of these approaches are quite effective for some of the singularities, they have the drawback that each of them only deals with one particular singularity. In this example, we present quadrature rules for the integration of functions of the form

$$\sum_{k=0}^{n} (\gamma_k \cdot log(x) + \sum_{j=1}^{m} \beta_{k,j} \cdot x^{\alpha_j}) \cdot P_k(x) \tag{50}$$

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[0, 1]$, $\beta_{k,j}$, γ_k are arbitrary real numbers, and α_j are arbitrary real numbers on the interval $[-0.6, 1]$.

Table 1: 16-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 4$, and precision 10^{-15}

x_i	w_i
0.1646476245461994E-18	0.2477997131959177E-17
0.2004881755033198E-13	0.1863311166024058E-12
0.4902407997203263E-10	0.3215991324579055E-09
0.1396853977847601E-07	0.6788563189534853E-07
0.9715236454504147E-06	0.3586206403622012E-05
0.2502196135803993E-04	0.7130636866829449E-04
0.3120851149673110E-03	0.6951436010759356E-03
0.2264576163994000E-02	0.3979838127986921E-02
0.1086917746927712E-01	0.1515746778330600E-01
0.3777218640280392E-01	0.4182483334409624E-01
0.1013279037973986E+00	0.8854031057518543E-01
0.2196196157836697E+00	0.1490380907486389E+00
0.3972680999338400E+00	0.2028312538451011E+00
0.6135562966157080E+00	0.2216836945000430E+00
0.8216868417553706E+00	0.1844567448110479E+00
0.9636466562372551E+00	0.9171766188102896E-01

In order to design such quadratures, we choose a reasonably large natural m, construct m Legendre nodes $\alpha_1, \alpha_2, \ldots, \alpha_m$, on the interval $[-0.6, 1]$, and use all functions of the forms

$$P_k(x) \cdot x^{\alpha_j}, \tag{51}$$

$$P_k(x) \cdot log(x) \tag{52}$$

as input functions ϕ_i for the algorithm of the preceding section. The result is a set quadratures for functions of the forms (51), (52). A somewhat involved analytical calculation shows that for sufficiently large m, the obtained quadratures will work for all functions of the form (50), and our numerical experiments show that $m = 100$ insures full double precision accuracy for all $\alpha_j \in [-0.6, 1]$.

In Tables 1 - 5, we list quadrature nodes and weights for $n = 4, 9, 19, 29$. In Tables 1, 3, 4, 5, the number of nodes is chosen to guarantee 15-digit accuracy. In Table 2, the number of nodes is chosen to guarantee 7 digits.

Example 5.2 The quadrature rules in this example are very similar to those in Example 5.1, except here we construct quadrature rules for functions singular at *both* ends of the interval where they are to be integrated. Specifically, integrands have the form

$$\sum_{k=0}^{n}\Big(\sum_{j=1}^{m}(a_{k,j} \cdot (1+x)^{\alpha_j} + b_{k,j} \cdot (1-x)^{\alpha_j}) + c_k \cdot log(1+x) + d_k \cdot log(1-x)\Big) \cdot P_k(x) \tag{53}$$

Table 4: 26-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 19$, and precision 10^{-15}

x_i	w_i
0.2852686209735951E-20	0.4390385492743041E-19
0.4655349788609637E-15	0.4445881189691443E-14
0.1432147899313873E-11	0.9689649973398580E-11
0.4915792345704672E-09	0.2471786670704959E-08
0.3986884553883893E-07	0.1527652265503579E-06
0.1168849078081257E-05	0.3470933550491954E-05
0.1630549221175312E-04	0.3803166416108812E-04
0.1307331567674635E-03	0.2422240257088061E-03
0.6884061227847875E-03	0.1022568448159836E-02
0.2620448293548410E-02	0.3143745934305781E-02
0.7740029188833982E-02	0.7549238041954824E-02
0.1872452403074940E-01	0.1495112040361046E-01
0.3869460001276389E-01	0.2548756008178511E-01
0.7058074961479188E-01	0.3865021281644121E-01
0.1165353335503884E+00	0.5342389042306681E-01
0.1775282580420220E+00	0.6849323863305738E-01
0.2531447462199369E+00	0.8243302008328313E-01
0.3415558481256653E+00	0.9386320384208941E-01
0.4396281348394975E+00	0.1015733726852001E+00
0.5431447278197111E+00	0.1046214551363520E+00
0.6471126706707170E+00	0.1024074963963311E+00
0.7461308154896283E+00	0.9472049436813551E-01
0.8347900655356778E+00	0.8175595131244442E-01
0.9080759999882411E+00	0.6410309004863602E-01
0.9617441758037388E+00	0.4270384642243640E-01
0.9926478556999123E+00	0.1881261305258270E-01

Table 5: 36-node quadrature for functions of the form (50), with $\alpha \in [-0.6, 1]$, $N = 39$, and precision 10^{-15}

x_i	w_i
0.1174238417413926E-19	0.1769042596381234E-18
0.1422439193737780E-14	0.1318732300270049E-13
0.3350676698582048E-11	0.2181187238172082E-10
0.8987762100979194E-09	0.4306047388907762E-08
0.5804062676082615E-07	0.2097251047066944E-06
0.1381879982602796E-05	0.3830347070073085E-05
0.1599014834456195E-04	0.3447814965093908E-04
0.1086072834052024E-03	0.1843012333973045E-03
0.4939690780979653E-03	0.6658876227138618E-03
0.1653457719227906E-02	0.1785581170381193E-02
0.4371083474213578E-02	0.3817614649487054E-02
0.9635942477742897E-02	0.6885390581283880E-02
0.1847241513238332E-01	0.1094085630140653E-01
0.3179190367214565E-01	0.1581728538518057E-01
0.5030636405050507E-01	0.2129142636454853E-01
0.7449442868952319E-01	0.2712481569656370E-01
0.1045979502135202E+00	0.3308456773919071E-01
0.1406326475828715E+00	0.3895216905306892E-01
0.1824044449022998E+00	0.4452688339606666E-01
0.2295280679235570E+00	0.4962723403902098E-01
0.2814468220422235E+00	0.5409202169130247E-01
0.3374533767644982E+00	0.5778135262022458E-01
0.3967116179369689E+00	0.6057773920656186E-01
0.4582796041927400E+00	0.6238718893653459E-01
0.5211335571597729E+00	0.6314016307782669E-01
0.5841926980689389E+00	0.6279229386348975E-01
0.6463446423449487E+00	0.6132477029316637E-01
0.7064709858680002E+00	0.5874432665998542E-01
0.7634726623238107E+00	0.5508279084756487E-01
0.8162946187294954E+00	0.5039617034177984E-01
0.8639493438008133E+00	0.4476327290202123E-01
0.9055387898384755E+00	0.3828387702474601E-01
0.9402742542357631E+00	0.3107648956468336E-01
0.9674938463383342E+00	0.2327578565976658E-01
0.9866773942995437E+00	0.1503024417658587E-01
0.9974613070359063E+00	0.6508977351752366E-02

Table 6: 22-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 4$, and precision 10^{-15}

$\pm x_i$	w_i
0.166008119316040E+00	0.3286464553329054E+00
0.4736467937561296E+00	0.2782402062916909E+00
0.7129463900017805E+00	0.1977249261400840E+00
0.8687173264995090E+00	0.1158087624474726E+00
0.9515411665787298E+00	0.5425992604604305E-01
0.9862971262509680E+00	0.1943874113675287E-01
0.9972429072629104E+00	0.4979788483749470E-02
0.9996464539418006E+00	0.8238003428108275E-03
0.9999757993153293E+00	0.7462712208720397E-04
0.9999993605804343E+00	0.2746237603563529E-05
0.9999999970230195E+00	0.2041880191195951E-07

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[-1, 1]$, $a_{k,j}$, $b_{k,j}$, c_k, d_k are arbitrary real numbers, and α_j are arbitrary real numbers on the interval $[-0.1, 1]$. Quadrature nodes and weights for $n = 4, 9, 19, 39$ are listed in Tables 6, 7, 8, 9 respectively; in all cases, the precision is 10^{-15}.

Example 5.3 In this example, we construct a direct generalization of quadratures constructed in Example 5.1, permitting the integrands to have power and logarithmic singularities *at arbitrary points on the closed half-line to the left of the interval of integration.* Specifically, integrands have the form

$$\sum_{k=0}^{n} (\gamma_k \cdot log(x+h) + \sum_{j=1}^{m} \beta_{k,j} \cdot (x+h)^{\alpha_j}) \cdot P_k(x) \tag{54}$$

where P_k denotes the (normalized) orthogonal polynomial of order k on the interval $[0, 1]$, $\beta_{k,j}$, γ_k are arbitrary real numbers, α_j are arbitrary real numbers on the interval $[-0.65, 1]$, and h is an arbitrary positive real number. In this case, the calculations were conducted in double precision; the 38-node quadrature formula for $n = 19$ is listed in Table 10; its precision is 10^{-9}.

Several observations can be made from the tables 1-8, and from the more detailed numerical experiments we have conducted.

- 1. The algorithm of this paper is always effective for Chebyshev systems; it almost always works for non-Chebyshev ones.
- 2. The scheme does not lose very many digits compared to the machine precision; when the calculations are performed in double precision, the quadratures can be obtained to 11 or 12 digits; the accuracy of quadratures in Tables 1-8 is full double precision; we used extended precision arithmetic in FORTRAN to obtain them.

Table 7: 27-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 9$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1969765126094452E+00
0.1953889665467211E+00	0.1922287111905558E+00
0.3814298736462841E+00	0.1784269782500965E+00
0.5496484616443740E+00	0.1568677485350913E+00
0.6932613279607421E+00	0.1296176364576521E+00
0.8078808016610349E+00	0.9937321489137896E-01
0.8920478424190657E+00	0.6925317917837661E-01
0.9475053154471952E+00	0.4247396818782292E-01
0.9790448975739819E+00	0.2179872525134398E-01
0.9936444652327659E+00	0.8672220251831163E-02
0.9986936386311707E+00	0.2388475528070173E-02
0.9998477986092101E+00	0.3837648653769931E-03
0.9999927156219827E+00	0.2671422777541431E-04
0.9999999335937359E+00	0.4068798910349743E-06

Table 8: 33-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 19$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1802406542699465E+00
0.1789856568226836E+00	0.1764865559769247E+00
0.3505713663705831E+00	0.1655482040246752E+00
0.5079970396268890E+00	0.1483733690643724E+00
0.6457344058749438E+00	0.1264620956535221E+00
0.7599840782344723E+00	0.1017484935648103E+00
0.8490304782768580E+00	0.7643386171408831E-01
0.9134021329241244E+00	0.5276203409291129E-01
0.9557717316319267E+00	0.3272086426808218E-01
0.9805181730564275E+00	0.1766845539228831E-01
0.9929045523533901E+00	0.7963812531655223E-02
0.9979798758935006E+00	0.2833884283485953E-02
0.9995837651123616E+00	0.7387521680930171E-03
0.9999445617386989E+00	0.1267394032662049E-03
0.9999960165362139E+00	0.1207609748958691E-04
0.9999998889650372E+00	0.4709227238502033E-06
0.9999999994557687E+00	0.3706639850258617E-08

Table 9: 45-node quadrature for functions of the form (53), with $\alpha \in [-0.1, 1]$, $N = 39$, and precision 10^{-15}

$\pm x_i$	w_i
0.0000000000000000E+00	0.1138212938786054E+00
0.1135283181390291E+00	0.1129431358863252E+00
0.2253080046824045E+00	0.1103317059272695E+00
0.3336364252858657E+00	0.1060558645237672E+00
0.4369024052356911E+00	0.1002294986469973E+00
0.5336306707891807E+00	0.9301028558331059E-01
0.6225248777667337E+00	0.8459812566475355E-01
0.7025089656717720E+00	0.7523338442881639E-01
0.7727667118189729E+00	0.6519506433099722E-01
0.8327794264993337E+00	0.5479889055074179E-01
0.8823615451977041E+00	0.4439489209928996E-01
0.9216930322777481E+00	0.3436308131973152E-01
0.9513451962287941E+00	0.2510376733595393E-01
0.9722913641056944E+00	0.1701539437521317E-01
0.9858845322639776E+00	0.1044852849794223E-01
0.9937724959340503E+00	0.5626146436355554E-02
0.9977200386244100E+00	0.2543352365327656E-02
0.9993454278943935E+00	0.9118380718941661E-03
0.9998636273258416E+00	0.2403706487446808E-03
0.9999815974719829E+00	0.4181929949775085E-04
0.9999986596740707E+00	0.4045883666118617E-05
0.9999999622133619E+00	0.1599158044436823E-06
0.9999999998137450E+00	0.1268296767711113E-08

Table 10: 38-node quadrature for functions of the form (54), with $\alpha \in [-0.65, 1]$, $N = 19$, and precision 10^{-9}

x_i	w_i
0.7629165866352161E-18	0.4643955333268610E-17
0.3799719398931375E-16	0.1132690565299208E-15
0.5684549949701512E-15	0.1423549582265871E-14
0.6085909916179373E-14	0.1371876219104025E-13
0.5277191865393953E-13	0.1094397021531007E-12
0.3900442913791902E-12	0.7534990994077416E-12
0.2535538557277294E-11	0.4603432835276850E-11
0.1481755662897140E-10	0.2545533729683496E-10
0.7911595380511587E-10	0.1293022088581050E-09
0.3907746000477183E-09	0.6102781198001779E-09
0.1803070816493823E-08	0.2700678436986190E-08
0.7833265344260583E-08	0.1128792193586090E-07
0.3224897189563689E-07	0.4482855569803782E-07
0.1264894823726299E-06	0.1700035548631482E-06
0.4747932260937661E-06	0.6182057321480894E-06
0.1711978528765632E-05	0.2163108715557027E-05
0.5948052018171647E-05	0.7302447810573277E-05
0.1995877304286260E-04	0.2382492261847977E-04
0.6475274273537152E-04	0.7511062044871306E-04
0.2029004100170709E-03	0.2279609908900293E-03
0.6109309950274235E-03	0.6592765068003472E-03
0.1747449285439932E-02	0.1781666222619331E-02
0.4661579935095226E-02	0.4378093849756735E-02
0.1135932523990354E-01	0.9537600800370288E-02
0.2491532030262493E-01	0.1820679046524441E-01
0.4902801284057732E-01	0.3060746663786768E-01
0.8713816071641225E-01	0.4600643316091537E-01
0.1415514175271372E+00	0.6292513465068938E-01
0.2128806314974303E+00	0.7951989233968431E-01
0.2998564528132552E+00	0.9391761648476182E-01
0.3994239415560721E+00	0.1044517799613406E+00
0.5070313867113639E+00	0.1098153664961849E+00
0.6170411438386144E+00	0.1091553255900476E+00
0.7232121752054713E+00	0.1021230666276667E+00
0.8192137516286219E+00	0.8888680524875885E-01
0.8991333728333283E+00	0.7010796674100402E-01
0.9579443204807173E+00	0.4688508195206744E-01
0.9919093183441774E+00	0.2069742637648333E-01

- 3. The algorithm of this paper is not very efficient. For example, the quadrature formula in Table 1 took about 2 minutes of CPU time on Ultra SPARC 2; the quadrature in Table 8 took about two hours of CPU time. Of course, extended precision on the UltraSparc is quite inefficient; in double precision, Table 8 took about 4 minutes on to construct. In any event, the quadratures of the type presented in this paper need not be constructed "on the fly"; the nodes and weights can be precomputed and stored. From this point of view, the CPU time requirements of our algorithm are not excessive. Still, its CPU time requirements grow as n^3 for large n, making it unsuitable for the construction of quadratures of very high order.

6 Generalizations and Conclusions

We have constructed a scheme for the design of Generalized Gaussian Quadratures for a fairly broad classes of functions. The results presented here should be viewed as somewhat experimental, since while the algorithm appears to work under quite general conditions, we can only *prove* that it *has to* work for Chebyshev systems.

Several possible extensions of the work suggest themselves.

1. While our numerical experiments indicate that the scheme of this paper works under very general conditions, we have only been able to prove that it has to work for Chebyshev systems (see Subsection 2.1 above). This discrepancy seems to indicate that it might be profitable to investigate generalizations of Theorem 2.1 to sets of functions other than Chebyshev systems.

2. By combining Observation 2.3 and Remark 2.4 with results in Sections 3, 4, it is fairly straightforward to construct algorithms for the efficient interpolation of fairly large classes of singular functions. For example, the nodes $x_1, x_2, \ldots, x_{36}$ in Table 5 lead to a stable interpolation formula on the interval $[0, 1]$ for all functions of the form

$$\sum_{k=0}^{n} P_k(x) \cdot \sum_{j=1}^{m} \beta_{k,j} \cdot x^{\alpha_j}, \tag{55}$$

with $-0.3 \leq \alpha_j \leq 1$, $0 \leq k \leq 19$, and the precision of interpolation 10^{-15}. Interpolation schemes of this type are currently under vigorous investigation, and will be reported in the near future.

3. In many situations (especially, in the numerical solution of partial differential equations), it is desirable to have "quadrature" formulae that, in addition evaluating integrals, would evaluate certain pseudodifferential operators, i.e. derivative, Hilbert Transform, derivative of the Hilbert Transform, etc. Clearly, such "quadratures" can not have positive weights, except for the Hilbert Transform. Several such quadratures have been constructed numerically, and the appropriate theory appears to be fairly straightforward; this work will be reported at a later date.

4. While the theory of Gaussian Quadratures in one dimension is extremely simple and well-understood, no similar theory exists in higher dimensions, except for a few scattered results (see, for example, [9],[14]). The approach of this paper is quite different from the classical

Gaussian Quadratures, and it appears possible to generalize it (at least, formally) to higher dimensions. While the advantages of such a construction would be significant, our investigation of it is at a very early stage. If successful, it will be reported at a later date.

References

[1] R. BULIRSCH, J. STOER, *Fehlerabschätzungen und Extrapolation mit Rationalen Funktionen bei Verfahren vom Richardson-Typus*, Numerische Mathematik, 6, 413-427 (1964).

[2] F. GANTMACHER AND M. KREIN, *Oscillation matrices and kernels and small oscillations of mechanical systems*, 2nd ed., Gosudarstv. Izdat. Tehn-Teor. Lit., Moscow, 1950 (Russian).

[3] W. GAUTCHI, *On Generating Orthogonal Polynomials*, SIAM Journal on Scientific and Statistical Computing, V. 3, No. 3, 1982.

[4] D. GOTTLIEB, S. ORSZAG, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia, 1977.

[5] T. HRYCAK, V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields*, SIAM Journal of Scientific Computing, Vol. 19, No. 6, pp. 1804-1826, 1998.

[6] S. KARLIN, *The Existence of Eigenvalues for Integral Operators*, Trans. Am. Math. Soc. v. 113, pp. 1-17 (1964).

[7] S. KARLIN, AND W. J. STUDDEN, *Tchebycheff Systems with Applications In Analysis And Statistics*, John Wiley (Interscience), New York, 1966.

[8] M. G. KREIN, *The Ideas of P. L. Chebyshev and A. A. Markov in the Theory Of Limiting Values Of Integrals*, American Mathematical Society Translations, Ser. 2, Vol. 12, 1959, pp. 1-122.

[9] J. N. LYNESS, D. JESPERSEN, *Moderate Degree Symmetric Quadrature Rules for the Triangle*, Journal of the Institute of Mathematics and its Applications, 1975, V. 15, pp. 19-32.

[10] J. MA, V. ROKHLIN, AND S. WANDZURA, *Generalized Gaussian Quadratures For Systems of Arbitrary Functions*, SIAM Journal of Numerical Analysis, v. 33, No. 3, pp. 971-996, 1996.

[11] A. A. MARKOV, *On the limiting values of integrals in connection with interpolation*, Zap. Imp. Akad. Nauk. Fiz.-Mat. Otd. (8) 6 (1898), no.5 (Russian), pp. 146-230 of [12].

[12] A. A. MARKOV, *Selected papers on continued fractions and the theory of functions deviating least from zero*, OGIZ, Moscow-Leningrad, 1948 (Russian).

[13] J. STOER, R. BULIRSCH, *Introduction to Numerical Analysis, Second Edition*, Springer-Verlag, 1993.

[14] S. WANDZURA, H. XIAO, *Quadrature Rules on Triangles in* $\mathbb{R}^2$, Yale University Technical Report YALEU/DCS/RR-1168 (1998).

[15] N. YARVIN AND V. ROKHLIN, *Generalized Gaussian Quadratures and Singular Value Decompositions of Integral Operators*, Yale University Technical Report YALEU/DCS/ RR-1109 (1996), to appear in *SIAM Journal on Scientific Computing.*

[16] N. YARVIN AND V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields on One-Dimensional Structures*, Yale University Technical Report YALEU/DCS/RR-1119 (1997), to appear in *SIAM Journal on Numerical Analysis.*

A Fast Adaptive Multipole Algorithm in Three Dimensions

H. Cheng, L. Greengard, V. Rokhlin
Research Report YALEU/DCS/RR-1158
August 3, 1998

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Ever since its introduction in the 1980's, the Fast Multipole Method has been capable of producing very high accuracy for an acceptable cost in two dimensions; in three dimensions, it has been considerably less efficient, except when the accuracy requirements were low. This situation changed somewhat with the appearance of a new version of the FMM in [12], which is highly efficient over a wide range of accuracies. That paper introduced a rather involved mathematical apparatus and described the algorithm in its simplest, non-adaptive form. In this paper, we describe an adaptive version of the scheme of [12], applicable to all distributions of particles that are likely to be encountered in practice. The performance of the algorithm is illustrated with numerical examples.

A Fast Adaptive Multipole Algorithm in Three Dimensions

H. Cheng, L. Greengard, V. Rokhlin
Research Report YALEU/DCS/RR-1158
August 3, 1998

The first and third authors were supported in part by DARPA/AFOSR under Contracts F49620-95-C-0075, F49620-97-1-0011. In addition, the third author was supported in part by ONR under Grant N00014-96-1-0188.
The second author was supported in part by the US Department of Energy under Contract DEFGO288ER25053 and in part by DARPA/AFOSR under Contract F49620-95-C-0075.
Approved for public release: distribution is unlimited.
Keywords: *Laplace Equation, Translation Operators, Fast Multipole Method, Adaptive Algorithms.*

A Fast Adaptive Multipole Algorithm in Three Dimensions

H. Cheng, L. Greengard and V. Rokhlin

August 3, 1998

1 Introduction

In [12], a new version of the Fast Multipole Method (FMM) for the evaluation of potential fields in three dimensions was introduced. The scheme evaluates all pairwise interactions in large ensembles of particles, i.e. expressions of the form

$$\Phi(x_j) = \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{q_i}{\|x_j - x_i\|} \tag{1}$$

for the gravitational or electrostatic potential and

$$E(x_j) = \sum_{\substack{i=1 \\ i \neq j}}^{n} q_i \cdot \frac{x_j - x_i}{\|x_j - x_i\|^3} \tag{2}$$

for the field, where $x_1, x_2, \cdots, x_n$ are points in $\mathbf{R}^3$, and $q_1, q_2, \cdots, q_n$ are a set of (real) coefficients.

The evaluation of expressions of the form (1) is closely related to a number of important problems in applied mathematics, physics, chemistry, and biology. These include molecular dynamics and quantum-mechanical simulations in chemistry, the evolution of large-scale gravitational systems in astrophysics, capacitance and inductance calculations in electrical engineering, and incompressible fluid dynamics. When certain closely related interactions are considered as well, involving expressions of the form

$$\Phi(x_j) = \sum_{\substack{i=1 \\ i \neq j}}^{n} q_i \cdot \frac{e^{i \cdot k \cdot \|x_j - x_i\|}}{\|x_j - x_i\|}, \tag{3}$$

the list of applications becomes even more extensive.

Ever since its introduction in the 1980's, the FMM has been capable of producing very high accuracy for an acceptable cost in two dimensions; in three dimensions, it has been considerably less efficient, except when the accuracy requirements were low. This situation changed somewhat with the development of a new version of the FMM in [12], which is highly efficient over a wide range of accuracies. That paper introduced a rather involved mathematical apparatus and described the algorithm in its simplest, non-adaptive form.

Needless to say, most charge distributions encountered in applications are highly non-uniform, and to be robust, a procedure for the evaluation of sums of the form (1) or (2) has to be adaptive. In this paper, we introduce such a scheme, applicable to all distributions of particles that are likely to be encountered in practice. An additional improvement introduced in this paper is a "compressed" version of translation operators used by the FMM procedure, which is the principal reason for the improvement of the timings found in Section 7 below over those in [12].

The paper is organized as follows. In Section 2, we summarize the mathematical and numerical facts to be used in subsequent sections. In Section 3, we review the analytical apparatus to be used in the design of the improved version of the FMM. Section 4 recapitulates the algorithm of [12], to be used as the starting point in the construction of the scheme of this paper. In Section 5, we describe the adaptive version of the FMM and make some comparisons with tree codes. In Section 6, we illustrate the performance of the method with several numerical examples. Finally, Section 7 discusses several possible generalizations. For a review of FMM-type methods and a more thorough discussion of the literature, we refer the reader to [12].

2 Mathematical preliminaries

In this section, we review the analytical tools used in the design of the FMM algorithm. For a detailed discussion, see [15, 14, 21, 9, 12].

We begin by defining the spherical harmonics of degree n and order m according to the formula

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} \cdot P_n^{|m|}(\cos\theta)e^{im\phi},. \tag{4}$$

Here, the special functions P_n^m are the associated Legendre functions, which can be defined by the Rodrigues' formula

$$P_n^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x),$$

where $P_n(x)$ denotes the Legendre polynomial of degree n.

Theorem 2.1 (Multipole Expansion). *Suppose that N charges of strengths $q_1, q_2, \cdots, q_N$ are located at points $X_1, X_2, ..., X_N$ with spherical coordinates $(\rho_1, \alpha_1, \beta_1)$, $(\rho_2, \alpha_2, \beta_2), \cdots,$ $(\rho_N, \alpha_N, \beta_N)$, respectively. Suppose further that the points $X_1, X_2, ..., X_N$ are located inside a sphere of radius a centered at the origin. Then, for any point $X = (r, \theta, \phi) \in \mathbb{R}^3$ with $r > a$, the potential $\Phi(X)$, generated by the charges $q_1, q_2, \cdots, q_N$, is given by the formula*

$$\Phi(X) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}} \cdot Y_n^m(\theta,\phi)\ , \tag{5}$$

where

$$M_n^m = \sum_{i=1}^{N} q_i \cdot \rho_i^n \cdot Y_n^{-m}(\alpha_i,\beta_i). \tag{6}$$

Furthermore, for any $p \geq 1$,

$$\left|\Phi(X) - \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} \cdot Y_n^m(\theta,\phi)\right| \left(\frac{\sum_{i=1}^{N} |q_i|}{r-a}\right)\left(\frac{a}{r}\right)^{p+1} . \tag{7}$$

The preceding theorem describes an efficient representation of the far field due to a collection of sources. Within the FMM, it is also useful to be able to describe the field locally when the charges themselves are far away.

Theorem 2.2 (Local Expansion) *Suppose that* N *charges of strengths* $q_1, q_2, \cdots, q_N$ *are located at the points* $X_1, X_2, \cdots, X_N$ *in* $\mathbb{R}^3$ *with spherical coordinates* $(\rho_1, \alpha_1, \beta_1)$, $(\rho_2, \alpha_2, \beta_2), \cdots,$ $(\rho_N, \alpha_N, \beta_N)$ *respectively. Suppose further that all the points* $X_1, X_2, \cdots, X_N$ *are located outside the sphere* S_a *of radius* a *centered at the origin. Then, for any point* $X \in S_a$ *with coordinates* (r, θ, ϕ), *the potential* $\Phi(X)$ *generated by the charges* $q_1, q_2, \cdots, q_N$ *is described by the local expansion:*

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta,\phi) \cdot r^j , \tag{8}$$

where

$$L_j^k = \sum_{l=1}^{N} q_l \cdot \frac{Y_j^{-k}(\alpha_l, \beta_l)}{\rho_l^{j+1}} , \tag{9}$$

with A_n^m *defined by (14). Furthermore, for any* $p \geq 1$,

$$\left|\Phi(X) - \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta,\phi) \cdot r^{j+1}\right| \leq \left(\frac{\sum_{i=1}^{N} |q_i|}{a-r}\right)\left(\frac{r}{a}\right)^{p+1} . \tag{10}$$

2.1 Translation Operators

The FMM relies on the ability to translate multipole and local expansions. The relevant translation operators are described in the next three theorems [11, 9].

Theorem 2.3 (Translation of a Multipole Expansion) *Suppose that* N *charges of strengths* $q_1, q_2, \cdots, q_N$ *are located inside the sphere* D *of radius* a *centered at* $X_0 = (\rho, \alpha, \beta)$. *Suppose further that for any point* $X = (r, \theta, \phi) \in \mathbb{R}^3 \setminus D$, *the potential due to these charges is given by the multipole expansion*

$$\Phi(X) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m}{r'^{n+1}} \cdot Y_n^m(\theta', \phi') , \tag{11}$$

where (r', θ', ϕ') *are the spherical coordinates of the vector* $X - X_0$.

Then, for any point $X = (r, \theta, \phi)$ outside a sphere D_1 of radius $(a + \rho)$ centered at the origin,

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} \cdot Y_j^k(\theta, \phi) \ , \tag{12}$$

where

$$M_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \frac{O_{j-n}^{k-m} \cdot i^{|k|-|m|-|k-m|} \cdot A_n^m \cdot A_{j-n}^{k-m} \cdot \rho^n \cdot Y_n^{-m}(\alpha, \beta)}{A_j^k} \ , \tag{13}$$

with A_n^m defined by the formula

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)! \cdot (n+m)!}} \ . \tag{14}$$

Furthermore, for any $p \geq 1$,

$$\left| \Phi(X) - \sum_{j=0}^{p} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} \cdot Y_j^k(\theta, \phi) \right| \leq \left(\frac{\sum_{i=1}^{N} |q_i|}{r - (a + \rho)} \right) \left(\frac{a + \rho}{r} \right)^{p+1} \ . \tag{15}$$

Definition 2.1 Formula (13) defines a linear operator converting the multipole expansion coefficients $\{O_j^k\}$ into the multipole expansion coefficients $\{M_j^k\}$. This linear mapping will be denoted by $\mathcal{T}_{MM}$.

Theorem 2.4 (Conversion of a Multipole Expansion to a Local Expansion) *Suppose that N charges of strengths $q_1, q_2, \cdots, q_N$ are located inside the sphere D_{X_0} of radius a centered at the point $X_0 = (\rho, \alpha, \beta)$, and that $\rho > (c + 1)a$ for some $c > 1$. Then the corresponding multipole expansion (11) converges inside the sphere D_0 of radius a centered at the origin. Furthermore, for any point $X \in D_0$ with coordinates (r, θ, ϕ), the potential due to the charges $q_1, q_2, \cdots, q_N$ is described by the local expansion:*

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \phi) \cdot r^j \ , \tag{16}$$

where

$$L_j^k = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m \cdot i^{|k-m|-|k|-|m|} \cdot A_n^m \cdot A_j^k \cdot Y_{j+n}^{m-k}(\alpha, \beta)}{(-1)^n \, A_{j+n}^{m-k} \cdot \rho^{j+n+1}} \ , \tag{17}$$

with A_n^m defined by (14). Furthermore, for any $p \geq 1$,

$$\left| \Phi(X) - \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \phi) \cdot r^j \right| \leq \left(\frac{\sum_{i=1}^{N} |q_i|}{ca - a} \right) \left(\frac{1}{c} \right)^{p+1} \ . \tag{18}$$

Definition 2.2 Formula (17) defines a linear operator converting the multipole expansion coefficients $\{O_j^k\}$ into the local expansion coefficients $\{L_j^k\}$. This linear mapping will be denoted by $\mathcal{T}_{ML}$.

Theorem 2.5 (Translation of a Local Expansion) *Suppose that X_0, X are a pair of points in $\mathbf{R}^3$ with spherical coordinates $(\rho, \alpha, \beta), (r, \theta, \phi)$ respectively, and (r', θ', ϕ') are the spherical coordinates of the vector $X - X_0$ and p is a natural number. Let X_0 be the center of a pth-order local expansion with p finite, its expression at the point X is given by the formula*

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m=-n}^{n} O_n^m \cdot Y_n^m(\theta', \phi') \cdot r'^n \ . \tag{19}$$

Then

$$\Phi(X) = \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \phi) \cdot r^j \ , \tag{20}$$

everywhere in $\mathbf{R}^3$, with

$$L_j^k = \sum_{n=j}^{p} \sum_{m=-n}^{n} \frac{O_n^m \cdot i^{|m|-|m-k|-|k|} \cdot A_{n-j}^{m-k} \cdot A_j^k \cdot Y_{n-j}^{m-k}(\alpha, \beta) \cdot \rho^{n-j}}{(-1)^{n+j} \cdot A_n^m}, \tag{21}$$

and A_n^m are defined by (14).

Definition 2.3 Formula (21) defines a linear operator converting the local expansion coefficients $\{O_n^m\}$ into the local expansion coefficients $\{L_n^m\}$. This linear mapping will be denoted by $\mathcal{T}_{LL}$.

Remark 2.1 The matrices representing the linear operators $\mathcal{T}_{MM}$, $\mathcal{T}_{ML}$, and $\mathcal{T}_{LL}$ are dense, so that applying them to truncated expansions with $O(p^2)$ coefficients costs $O(p^4)$ operations. This is one of principal reasons for the relatively high CPU time requirements of most existing FMM implementations in three dimensions. Section 3 of this paper provides tools for the rapid application of the operators $\mathcal{T}_{MM}, \mathcal{T}_{ML}, \mathcal{T}_{LL}$ to arbitrary vectors, improving the efficiency of FMM algorithms significantly.

2.2 Rotation Operators

In this subsection, we introduce operators which transform multipole and local expansions under rotations of the coordinate system. These operators will play a role in Section 3. The basic results are contained in the next two theorems, whose proofs can be found in [3], together with formulae for the evaluation of the coefficients $R_n^{m,m'}$ in (22), (23).

Theorem 2.6 (Rotation of Multipole Expansions) *Suppose that (e_1, e_2, e_3) are the three standard orthonormal basis vectors in $\mathbb{R}^3$, given by the formulae*

$$\begin{aligned} e_1 &= (1,0,0), \\ e_2 &= (0,1,0), \\ e_3 &= (0,0,1), \end{aligned}$$

and $(\omega_1, \omega_2, \omega_3)$ are three other orthonormal verctors in $\mathbb{R}^3$, forming another basis.

Suppose further that a harmonic function $\Phi : \mathbb{R}^3 \setminus \{0\} \mapsto \mathbb{R}$ is defined by the formula

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} \cdot Y_n^m(\theta, \phi),$$

with (r, θ, ϕ) the spherical coordinates of the point $X \in \mathbb{R}^3$ associated with the basis (e_1, e_2, e_3). Then, there exist coefficients $R_n^{m,m'}$ with $n = 0, 1, \cdots, p$, $m = -n, ..., n$, $m' = -n, ..., n$, such that for any $X \in \mathbb{R}^3$,

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m'=-n}^{n} \frac{\tilde{M}_n^{m'}}{r^{n+1}} \cdot Y_n^{m'}(\theta', \phi'),$$

where (r, θ', ϕ') are spherical coordinates of X in the system of coordinates associated with the basis $(\omega_1, \omega_2, \omega_3)$, and

$$\tilde{M}_n^{m'} = \sum_{m=-n}^{n} R_n^{m,m'} \cdot M_n^m, \tag{22}$$

for all $n = 0, 1, ..., p$, $m' = -n, ..., n$.

Theorem 2.7 (Rotation of Local Expansions) *Under the conditions of Theorem 2.6, suppose that a harmonic function $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}$ is defined by the formula*

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m=-n}^{n} L_n^m \cdot r^{n+1} \cdot Y_n^m(\theta, \phi),$$

where (r, θ, ϕ) are the spherical coordinates of the point $X \in \mathbb{R}^3$ associated with the basis (e_1, e_2, e_3). Then for any $X \in \mathbb{R}^3$,

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m'=-n}^{n} \tilde{L}_n^{m'} \cdot r^{n+1} \cdot Y_n^{m'}(\theta', \phi'),$$

where (r, θ', ϕ') are spherical coordinates of X in the system of coordinates associated with the basis $(\omega_1, \omega_2, \omega_3)$, and

$$\tilde{L}_n^{m'} = \sum_{m=-n}^{n} R_n^{m,m'} \cdot L_n^m, \tag{23}$$

for all $n = 0, 1, ..., p$, $m' = -n, ..., n$. Furthermore, the coefficients $R_n^{m,m'}$ are the same as in (22).

Definition 2.4 Given a rotation $\Omega : \mathbf{R}^3 \mapsto \mathbf{R}^3$, formulae (22), (23) define operators converting the multipole coefficients $\{M_n^m\}$ into the multipole coefficients $\{\bar{M}_n^m\}$ and the local coefficients $\{L_n^m\}$ into the local coefficients $\{\bar{L}_n^m\}$, respectively. These two operators are identical, and will be denoted by $\mathcal{R}(\Omega)$.

Remark 2.2 An inspection of formulae (22), (23) shows immediately that the numerical evaluation of the operator $\mathcal{R}(\Omega)$ requires $O(p^3)$ operations.

2.3 Exponential representation

The new generation of FMMs is based on a combination of multipole expansions and exponential or "plane wave" expansions. Given a source point $P = (x_0, y_0, z_0)$ and a target location $Q = (x, y, z)$, with $z > z_0$ and $r = \|P - Q\|$, we begin with the formula [16]

$$\frac{1}{r} = \frac{1}{2\pi} \int_0^\infty e^{-\lambda(z-z_0)} \int_0^{2\pi} e^{i\lambda((x-x_0)\cos\alpha + (y-y_0)\sin\alpha)} d\alpha \, d\lambda. \tag{24}$$

We will construct approximations to the integral in (24) via appropriately chosen quadrature formulae. These quadratures are investigated in detail in [23]; in the following lemma, we simply state the result for three special cases, corresponding to three-digit, six-digit and nine-digit accuracy.

Lemma 2.8 ([23, 12]) *Suppose that $X_0 = (x_0, y_0, z_0)$, $X = (x, y, z)$ are a pair of points in $\mathbf{R}^3$, and that $r = \|X - X_0\|$. Suppose further that the coordinates $(x - x_0, y - y_0, z - z_0)$ of the vector $X - X_0$ satisfy the conditions*

$$1 \le z - z_0 \le 4, \quad 0 \le \sqrt{(x-x_0)^2 + (y-y_0)^2} \le 4\sqrt{2}. \tag{25}$$

Then

$$\left| \frac{1}{r} - \sum_{k=1}^{8} \frac{w_k^3}{M_k^3} \sum_{j=1}^{M_k^3} e^{-\lambda_k^3 \cdot [(z-z_0) - i(x-x_0)\cdot\cos(\alpha_{j,k}^3) - (y-y_0)\cdot\sin(\alpha_{j,k}^3)]} \right| < 1.6 \times 10^{-3}, \tag{26}$$

$$\left| \frac{1}{r} - \sum_{k=1}^{17} \frac{w_k^6}{M_k^6} \sum_{j=1}^{M_k^6} e^{-\lambda_k^6 \cdot [(z-z_0) - i(x-x_0)\cdot\cos(\alpha_{j,k}^6) - (y-y_0)\cdot\sin(\alpha_{j,k}^6)]} \right| < 1.3 \times 10^{-6}, \tag{27}$$

$$\left| \frac{1}{r} - \sum_{k=1}^{26} \frac{w_k^9}{M_k^9} \sum_{j=1}^{M_k^9} e^{-\lambda_k^9 \cdot [(z-z_0) - i(x-x_0)\cdot\cos(\alpha_{j,k}^9) - (y-y_0)\cdot\sin(\alpha_{j,k}^9)]} \right| < 1.1 \times 10^{-9}, \tag{28}$$

where $\alpha_{j,k}^3 = 2\pi j / M_k^3$, $\alpha_{j,k}^6 = 2\pi j / M_k^6$, $\alpha_{j,k}^9 = 2\pi j / M_k^9$. The weights $\{w_l^3, l = 1, ..., 8\}$, $\{w_l^6, l = 1, ..., 17\}$, $\{w_l^9, l = 1, ..., 26\}$, the nodes $\{\lambda_l^3, l = 1, ..., 8\}$, $\{\lambda_l^6, l = 1, ..., 17\}$, $\{\lambda_l^9, l = 1, ..., 26\}$ and the integer arrays $\{M_k^3, k = 1, ..., 8\}$, $\{M_k^6, k = 1, ..., 17\}$, $\{M_k^9, k = 1, ..., 26\}$ are given in Tables 13, 14, 15 of the Appendix, respectively.

Remark 2.3 The conditions (25) in the preceding Lemma appear to be rather special. They are, however, related to the geometric refinement of space introduced by the FMM and their use will become clear in the next section.

Remark 2.4 When the desired precision is clear from the context, we will simplify the notation used in Lemma 2.8, writing each of the expressions (26), (27), (28) in the form

$$\left| \frac{1}{r} - \sum_{k=1}^{s(\varepsilon)} \frac{w_k}{M_k} \sum_{j=1}^{M_k} e^{-\lambda_k \cdot (z-z_0)} \cdot e^{i\lambda_k \cdot [(x-x_0)\cdot \cos(\alpha_{j,k}) + (y-y_0)\cdot \sin(\alpha_{j,k})]} \right| < \varepsilon, \tag{29}$$

where the integers $s(\varepsilon)$ and the triplets $\{M_k, w_k, \lambda_k |\ k = 1, \ldots, \varepsilon\}$ all depend on ε, and $\alpha_{j,k} = 2\pi j / M_k$. The total number of exponential basis functions used in (29) will be denoted by

$$S_{exp} = \sum_{k=1}^{s(\varepsilon)} M_k. \tag{30}$$

3 Data Structures and Fast Translation Operators

In order to develop a fast algorithm, we first define the computational domain to be the smallest cube in $\mathbf{R}^3$ containing all sources. We then build a hierarchy of boxes, refining the computational domain into smaller and smaller regions. At refinement level 0, we have a single box corresponding to the entire computational domain. Refinement level $l + 1$ is obtained recursively from level l by the subdivision of each box into eight cubic boxes of equal size. In the nonadaptive case, this recursive process is halted after roughly $\log_8 N$ levels, where N is the total number of sources under consideration.

Definition 3.1 A box c is said to be a *child* of box b, if box c is obtained by a single subdivision of box b. Box b is said to be the *parent* of box c.

Definition 3.2 Two boxes are said to be *colleagues* if they are at the same refinement level and share a boundary point. (A box is considered to be a colleague of itself.) The set of colleagues of a box b will be denoted by $Coll(b)$.

Definition 3.3 Two boxes are said to be *well separated* if they are at the same refinement level and are not colleagues.

Definition 3.4 With each box b is associated an *interaction list*, consisting of the children of the colleagues of b's parent which are well separated from box b (Figure 1).

Note that a box can have up to 27 colleagues and that its interaction list contains up to 189 boxes. Figure 1 depicts the colleagues and interaction list of a box in a two-dimensional setting.

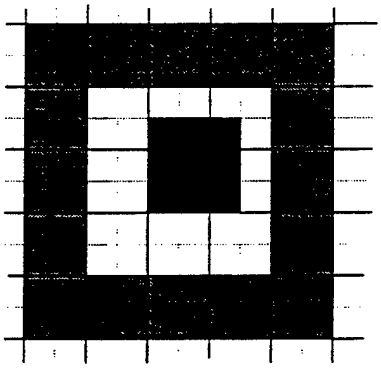

Figure 1: The colleagues of a (two-dimensional) box b are darkly shaded, while its interaction list is indicated in white. In three dimensions, a box b has up to 27 colleagues and its interaction list contains up to 189 boxes.

The interaction list for each box will be further subdivided into six lists, associated with the six coordinate directions $(+z, -z, +y, -y, +x, -x)$ in the three dimensional coordinate system. We will refer to the $+z$ direction as *up*, the $-z$ direction as *down*, the $+y$ direction as *north*, the $-y$ direction as *south*, the $+x$ direction as *east*, and the $-x$ direction as *west*.

Definition 3.5 (Directional lists)

The *Uplist* for a box b consists of those elements of the interaction list which lie *above* b and are separated by at least one box in the $+z$-direction (Fig. 2).

The *Downlist* for a box b consists of those elements of the interaction list which lie *below* b and are separated by at least one box in the $-z$-direction.

The *Northlist* for a box b consists of those elements of the interaction list which lie *north* of b, are separated by at least one box in the $+y$-direction, and are not contained in the Up or Down lists.

The *Southlist* for a box b consists of those elements of the interaction list which lie *south* of b, are separated by at least one box in the $-y$-direction, and are not contained in the Up or Down lists.

The *Eastlist* for a box b consists of those elements of the interaction list which lie *east* of b, are separated by at least one box in the $+x$-direction, and are not contained in the Up, Down, North, or South lists.

The *Westlist* for a box b consists of those elements of the interaction list which lie west of b, are separated by at least one box in the $-x$-direction, and are not contained in the *Up*, *Down*, *North*, or *South* lists.

For any box b, we will denote the number of elements in its *Uplist* by $N(Uplist(b))$, and adopt a similar convention for each of the remain five lists.

Remark 3.1 It is easy to verify that the original interaction list is equal to the union of the

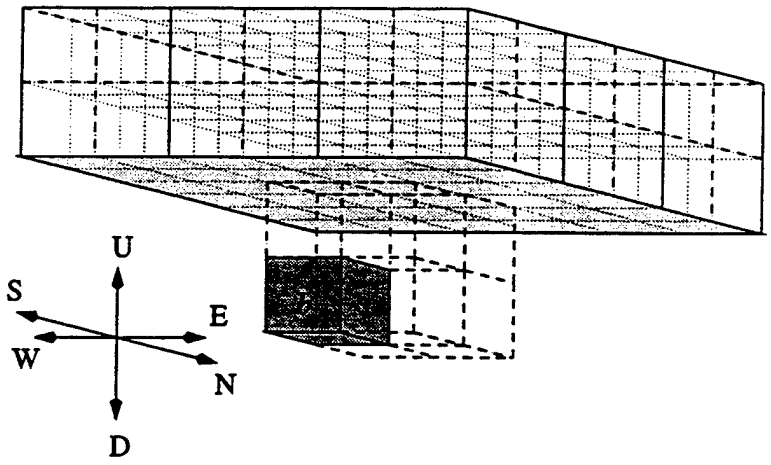

Figure 2: The *Uplist* for the box b (see Definition 3.5).

Up, *Down*, *North*, *South*, *East* and *West* lists. It is also easy to verify for two boxes b, c that

$$\begin{aligned} c \in \mathit{Uplist}(b) &\Leftrightarrow b \in \mathit{Downlist}(c), \\ c \in \mathit{Northlist}(b) &\Leftrightarrow b \in \mathit{Southlist}(c), \\ c \in \mathit{Eastlist}(b) &\Leftrightarrow b \in \mathit{Westlist}(c). \end{aligned} \tag{31}$$

Furthermore, suppose that two boxes b and c are of unit volume and that $c \in \mathit{Uplist}(b)$. Then for any point $X_0 = (x_0, y_0, z_0) \in b$ and any point $X = (x, y, z) \in c$, the vector $X - X_0 = (x - x_0, y - y_0, z - z_0)$ satisfies the inequality

$$1 \leq z - z_0 \leq 4, \;\; 0 \leq \sqrt{(x - x_0)^2 + (y - y_0)^2} \leq 4\sqrt{2}. \tag{32}$$

Note that this is precisely the condition (25) in Lemma 2.8.

Remark 3.2 When there is no danger of confusion, we will use *Uplist(b)* to refer to the geometrical region defined by the *union* of all boxes in the *Uplist* of box b. This is a slight abuse of notation, since *Uplist(b)* is, strictly speaking, a *set* of boxes. We will take the same liberty with *Downlist(b)*, *Northlist(b)*, *Southlist(b)*, *Eastlist(b)*, *Westlist(b)* and $\mathit{Coll}(b)$.

3.1 Rotation Based Translation Operators

In this section, we describe a simple scheme for reducing the cost of applying any of the three operators $\mathcal{T}_{MM}, \mathcal{T}_{ML}, \mathcal{T}_{LL}$ to an arbitrary vector from $O(p^4)$ to $O(p^3)$ operations. The scheme is based on the observation that when a multipole or local expansion is translated along the z-axis, the cost is reduced from $O(p^4)$ to $O(p^3)$ [5, 12, 22]. The following lemma is obtained immediately from the resulting simplification of formulae (13), (17) and (21).

Lemma 3.1 *If, in Theorems 2.3, 2.4 and 2.5, the spherical coordinates of the point X_0 are $(\rho, 0, 0)$, then the formulae (13), (17) and (21) assume the form*

$$M_j^k = \sum_{n=0}^{j} \frac{O_{j-n}^k \cdot A_n^0 \cdot A_{j-n}^k \cdot \rho^n \cdot Y_n^0(0,0)}{A_j^k}, \tag{33}$$

$$L_j^k = \sum_{n=0}^{\infty} \frac{O_n^k \cdot A_n^k \cdot A_j^k \cdot Y_{j+n}^0(0,0)}{(-1)^n A_{j+n}^0 \cdot \rho^{j+n+1}}, \tag{34}$$

$$L_j^k = \sum_{n=j}^{p} \frac{O_n^k \cdot A_{n-j}^0 \cdot A_j^k \cdot Y_{n-j}^0(0,0) \cdot \rho^{n-j}}{(-1)^{n+j} \cdot A_n^k}, \tag{35}$$

respectively.

Definition 3.6 The special cases of the linear operators $\mathcal{T}_{MM}$, $\mathcal{T}_{ML}$, and $\mathcal{T}_{LL}$ defined by the formulae (33), (34), and (35) will be denoted by $\mathcal{T}_{MM}^z$, $\mathcal{T}_{ML}^z$, and $\mathcal{T}_{LL}^z$ respectively.

Observation 3.3 (Rotation Based Translation Operators) Inspection of formulae (33), (34), (35) indicates that each of the operators $\mathcal{T}_{MM}^z$, $\mathcal{T}_{ML}^z$ and $\mathcal{T}_{LL}^z$ can be applied numerically to an arbitrary pth-order expansion for a cost proportional to p^3. Thus, a translation operator can be applied to an arbitrary vector for a cost proportional to p^3 via the following procedure. First, the system of coordinates is rotated so that the new z-axis points to the desired translation center. Then, the expansion is translated via one of the formulae (33), (34) and (35). Finally, the translated expansion is rotated back to the original system of coordinates. Since each of the three stages costs $O(p^3)$ operations, the cost of the whole process has also been reduced to $O(p^3)$ operations. Formally, the scheme we have outlined corresponds to the factorizations

$$\mathcal{T}_{MM} = \mathcal{R}(\Omega^{-1}) \circ \mathcal{T}_{MM}^z \circ \mathcal{R}(\Omega), \tag{36}$$

$$\mathcal{T}_{ML} = \mathcal{R}(\Omega^{-1}) \circ \mathcal{T}_{ML}^z \circ \mathcal{R}(\Omega), \tag{37}$$

$$\mathcal{T}_{LL} = \mathcal{R}(\Omega^{-1}) \circ \mathcal{T}_{LL}^z \circ \mathcal{R}(\Omega), \tag{38}$$

where $\mathcal{R}(\Omega)$ is defined in section 2.2 and $\mathcal{R}(\Omega^{-1})$ denotes the inverse rotation operator.

3.2 Plane Wave Based Translation Operators

In three-dimensional fast multipole schemes, the operator $\mathcal{T}_{ML}$ (converting multipole expansions into local ones) tends to be applied much more frequently then the operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$ which shift multipole and local expansions. Ignoring boundary effects, one ends up applying $\mathcal{T}_{ML}$ to the multipole expansion for each box about 189 times when the charge distribution is uniform. The operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$, on the other hand, are applied roughly once per box. In the algorithm of this paper, the operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$ are applied via the order p^3 scheme described in the preceding section; $\mathcal{T}_{ML}$ is applied by means of a much more complicated procedure, involving the plane wave representation introduced in on Lemma 2.8 of section 2.3.

The following observation provides an expansion of the form (29) for the potential generated by a collection of charges. It is an immediate consequence of Lemma 2.8.

Observation 3.4 Suppose that N charges of strengths $q_1, q_2, ..., q_N$ are located at points $X_1, X_2, \cdots, X_N$ in $\mathbb{R}^3$ with Cartesian coordinates $(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_N, y_N, z_N)$, respectively. Suppose further that all points $X_1, X_2, ..., X_N$ are inside a cubic box b with unit volume centered at the origin and that $X = (x, y, z) \in \mathbb{R}^3$ such that $X \in Uplist(b)$. Let $\Phi(X)$ denote the potential generated by the charges $q_1, q_2, ..., q_N$ and let Ψ_ϵ be defined by the formula

$$\Psi_\epsilon(X) = \sum_{k=1}^{s(\epsilon)} \sum_{j=1}^{M_k} W(k,j) \cdot e^{-\lambda_k z} \cdot e^{i\lambda_k \cdot (x \cdot \cos(\alpha_{j,k}) + y \cdot \sin(\alpha_{j,k}))}, \tag{39}$$

with the coefficients $W(k,j)$ given by the formula

$$W(k,j) = \frac{w_k}{M_k} \sum_{l=1}^{N} q_i \cdot e^{\lambda_k z_l} \cdot e^{-i\lambda_k \cdot (x_l \cdot \cos(\alpha_{j,k}) + y_l \cdot \sin(\alpha_{j,k}))}, \tag{40}$$

for all $k = 1, ..., s(\epsilon), j = 1, ..., M_k$. Then, if $A = \sum_{l=1}^{N} |q_l|$, we have the estimate

$$|\Phi(X) - \Psi_\epsilon(X)| < A\,\epsilon. \tag{41}$$

Observation 3.5 A somewhat involved analysis shows that, under the conditions of the preceding observation, $s(\epsilon) \sim p$, where p is chosen according to (7) to achieve the same accuracy using a multipole expansion. Likewise, the total number of exponential basis functions S_{exp} in (39) is of the same order as the total number of multipole moments (p^2) in (7) in order that the two expansions provide the same precision ϵ.

Expansions of the form (39) will be referred to as *exponential expansions.* Their main utility is that translation takes a particularly simple form.

Theorem 3.2 (Diagonal translation) *Suppose that a function $\Psi_\epsilon(X) : \mathbb{R}^3 \mapsto \mathbb{C}$ is defined by the formula (39), which we view as an expansion centered at the origin for $X = (x, y, z)$. Then, for any vector $X_0 = (x_0, y_0, z_0) \in \mathbb{R}^3$, we have the shifted expansion*

$$\Psi_\epsilon(X) = \sum_{k=1}^{s(\epsilon)} \sum_{j=1}^{M_k} V(k,j) \cdot e^{-\lambda_k (z - z_0)} \cdot e^{i\lambda_k \cdot ((x-x_0) \cdot \cos(\alpha_{j,k}) + (y-y_0) \cdot \sin(\alpha_{j,k}))}, \tag{42}$$

where

$$V(k,j) = W(k,j) \cdot e^{-\lambda_k z_0} \cdot e^{i\lambda_k \cdot (x_0 \cdot \cos(\alpha_{j,k}) + y_0 \cdot \sin(\alpha_{j,k}))}, \tag{43}$$

for $k = 1, ..., s(\epsilon)$, $j = 1, ..., M_k$.

Definition 3.7 Formula (43) defines a linear operator mapping the coefficients $\{W(k,j)\}$ to the coefficients $\{V(k,j)\}$. This linear operator will be denoted by $\mathcal{D}_{exp}$.

The operator $\mathcal{D}_{exp}$ provides a tool for translating expansions of the form (39) at a cost of $O(S_{exp}) \sim O(p^2)$ operations. In FMM algorithms, however, it is convenient to be able to use multipole and local expansions. Thus, in order to be able to use the operator $\mathcal{D}_{exp}$, linear operators converting multipole expansions into exponential expansions and exponential expansions into local expansions have to be constructed. The following two theorems provide such operators.

Theorem 3.3 *Suppose that N charges of strengths $q_1, q_2, \cdots, q_N$ are located inside a box b of volume d^3 centered at the origin, ε is a positive real number and p is an integer such that for any point $X \in$ Uplist(b) with spherical coordinates (r, θ, ϕ), the potential $\Phi(X)$ generated by the charges $q_1, q_2, \cdots, q_N$ satisfies the inequality*

$$\left| \Phi(X) - \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{O_n^m}{r^{n+1}} \cdot Y_n^m(\theta, \phi) \right| < \varepsilon. \tag{44}$$

Then

$$\left| \Phi(X) - \sum_{k=1}^{s(\varepsilon)} \sum_{j=1}^{M_k} W(k,j) \cdot e^{-(\lambda_k/d)\cdot z} \cdot e^{i(\lambda_k/d)\cdot(x\cdot\cos(\alpha_{j,k})+y\cdot\sin(\alpha_{j,k}))} \right| < (A/d+1)\cdot\varepsilon, \tag{45}$$

where (x, y, z) are the Cartesian coordinates of X, $A = \sum_{l=1}^{N} |q_l|$, and

$$W(k,j) = \frac{w_k/d}{M_k} \sum_{m=-p}^{p} (-i)^{|m|} \cdot e^{im\cdot\alpha_{j,k}} \sum_{n=|m|}^{p} \frac{O_n^m}{\sqrt{(n-m)!(n+m)!}} (\lambda_k/d)^n, \tag{46}$$

for $k = 1, ..., s(\varepsilon)$, $j = 1, ..., M_k$.

Definition 3.8 Formula (46) defines a linear operator converting the coefficients $\{O_n^m\}$ into the coefficients $\{W(k,j)\}$. This linear mapping will be denoted by C_{MX}.

Theorem 3.4 *Suppose that N charges of strengths $q_1, q_2, \cdots, q_N$ are located inside a box b of volume d^3 centered at the origin, ε is a positive real number, and that for any point $X = (x, y, z) \in$ Uplist(b), the potential $\Phi(X)$ generated by the charges $q_1, q_2, \cdots, q_N$ satisfies the inequality*

$$\left| \Phi(X) - \sum_{k=1}^{s(\varepsilon)} \sum_{j=1}^{M_k} W(k,j) \cdot e^{-(\lambda_k/d)\cdot z} \cdot e^{i(\lambda_k/d)\cdot(x\cdot\cos(\alpha_{j,k})+y\cdot\sin(\alpha_{j,k}))} \right| < (A/d)\cdot\varepsilon, \tag{47}$$

where $A = \sum_{l=1}^{N} |q_l|$. Then there exists an integer p, such that

$$\left| \Phi(X) - \sum_{n=0}^{p} \sum_{m=-n}^{n} L_n^m \cdot Y_n^m(\theta, \phi) \cdot r^n \right| < (A/d+1)\cdot\varepsilon, \tag{48}$$

where (r, θ, ϕ) are the spherical coordinates of X and

$$L_n^m = \frac{(-i)^{|m|}}{\sqrt{(n-m)!(n+m)!}} \sum_{k=1}^{s(\varepsilon)} (-\lambda_k/d)^n \sum_{j=1}^{M_k} W(k,j) \cdot e^{im\cdot\alpha_{j,k}}, \tag{49}$$

for $n = 0, ..., p$, $m = -n, ..., n$.

Definition 3.9 Formula (49) defines a linear operator converting the coefficients $\{W(k,j)\}$ into the coefficients $\{L_n^m\}$. This linear mapping will be denoted by $\mathcal{C}_{XL}$.

Remark 3.6 It is easy to see that (46) can be evaluated numerically for $k = 1, ..., s(\varepsilon), j = 1, ..., M_k$, at a cost proportional to p^3. Indeed, we first calculate $(2p+1) \cdot s(\varepsilon)$ quantities $F_{k,m}$ defined by the formula

$$F_{k,m} = \sum_{n=|m|}^{p} \frac{O_n^m}{\sqrt{(n-m)!(n+m)!}} (\lambda_k/d)^n, \tag{50}$$

for $k = 1, ..., s(\varepsilon), m = -p, ..., p$. This step requires $O(s(\varepsilon) \cdot p^2)$ operations. We then evaluate the coefficients $W(k,j)$ via the formula

$$W(k,j) = \frac{w_k/d}{M_k} \sum_{m=-p}^{p} (-i)^{|m|} \cdot e^{im \cdot \alpha_{j,k}} \cdot F_{k,m}, \tag{51}$$

for $k = 1, ..., s(\varepsilon)$, $j = 1, ..., M_k$, at a cost of $O(S_{exp} \cdot p)$ operations. Thus, the total cost of applying the operator $\mathcal{C}_{MX}$ numerically to a pth-order multipole expansion is

$$Cost(\mathcal{C}_{MX}) \sim O(p^2 s(\varepsilon) + pS_{exp}) \sim O(p^3), \tag{52}$$

making use of Observation 3.5. A similar argument shows that the operator $\mathcal{C}_{XL}$ can also be evaluated numerically for a cost proportional to p^3.

The proofs of Theorems 3.2, 3.3, 3.4 can be found in [12]. The following observation follows immediately from Theorems 3.2, 3.3 and 3.4.

Observation 3.7 (Multipole to local translation for the Uplist) Suppose that b, c are two boxes such that c is in the *Uplist* of b. Then the translation operator $\mathcal{T}_{ML}$ which converts a multipole expansion centered in b to a local expansion centered in c can be applied via the following procedure. First, convert the multipole expansion centered in b into an exponential expansion via the operator $\mathcal{C}_{MX}$; then, use the operator $\mathcal{D}_{exp}$ to translate the resulting exponential expansion to the center of box c; finally, convert the latter expansion into a local expansion in box c via the operator $\mathcal{C}_{XL}$. In short,

$$\mathcal{T}_{ML} = \mathcal{C}_{XL} \circ \mathcal{D}_{exp} \circ \mathcal{C}_{MX}. \tag{53}$$

Observation 3.8 (Multipole to local translation: general case) The decomposition (53) of the operator $\mathcal{T}_{ML}$ is valid only when box c is in the *Uplist* of box b. When box c is not in the *Uplist* of box b, the operator $\mathcal{T}_{ML}$ can easily be applied by first rotating the system of coordinates, so that in the new coordinate system, box c lies in the *Uplist* of box b, applying the operator $\mathcal{T}_{ML}$ via (53) to the rotated expansion, and finally rotating back to the original system of coordinates. Formally, this corresponds to the factorization

$$\mathcal{T}_{ML} = \mathcal{R}(\Omega^{-1}) \circ \mathcal{C}_{XL} \circ \mathcal{D}_{exp} \circ \mathcal{C}_{MX} \circ \mathcal{R}(\Omega). \tag{54}$$

The rotation operators $\mathcal{R}(\Omega)$ are described in section 2.2.

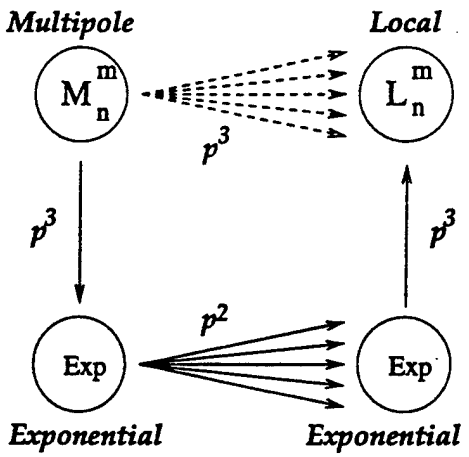

Figure 3: A large number of multipole-to-local translations, each costing $O(p^3)$ operations are replaced by a single multipole-to-exponential operator costing $O(p^3)$ operations, a large number of exponential translations costing $O(p^2)$ operations, and a single exponential-to-local operator costing $O(p^3)$ operations.

Remark 3.9 As mentioned earlier, application of the translation operators $\mathcal{T}_{ML}$ is a dominant part of FMM algorithms, occurring up to 189 times per box. Naive application of these operators results in a cost of roughly $189 \cdot p^4$ operations per box, which is prohibitively expensive in most cases. Fast rotation-based schemes [5, 22, 12] use Observation 3.3 to reduce the cost to roughly $189 \cdot 3 \cdot p^3$ operations per box; the resulting FMM schemes are fairly efficient in low-precision applications. Theorems 3.2, 3.3, 3.4 of this subsection can be used to reduce the cost of application of the operators $\mathcal{T}_{ML}$ to approximately $20 \cdot p^3 + 189 \cdot p^2$ operations per box. Indeed, in order to account for the interaction of box b with its *Uplist* boxes, we use the operator $\mathcal{C}_{MX}$ of Theorem 3.3 to convert b's multipole expansion into an exponential one for a cost proportional to p^3. We then use the operator $\mathcal{D}_{exp}$ of Theorem 3.2 to translate the resulting exponential expansion to each of the boxes in *Uplist(b)*, for a cost propotional to $N(\mathit{Uplist}(b)) \cdot p^2$. Subsequently, we convert the accumulated exponential expansion for each box into a local one via the operator $\mathcal{C}_{XL}$ of Theorem 3.4, for a cost proportional to p^3. This procedure is illustrated in Figure 3. The analogous process must, of course, be repeated for the *Downlist, Northlist, Southlist, Eastlist*, and *Westlist*. For the *Northlist, Southlist, Eastlist*, and *Westlist* (but not for the *Downlist*), there is an additional cost proportional to $2 \cdot p^3$ operations per box to rotate the coordinate system, as described in Observation 3.8. The total cost for each of the six interaction lists is summarized in the following

$$\begin{aligned} \mathit{Cost}(\mathit{Uplist}) &\sim 2 \cdot p^3 + N(\mathit{Uplist}(b)) \cdot p^2, \\ \mathit{Cost}(\mathit{Downlist}) &\sim 2 \cdot p^3 + N(\mathit{Downlist}(b)) \cdot p^2, \\ \mathit{Cost}(\mathit{Northlist}) &\sim 4 \cdot p^3 + N(\mathit{Northlist}(b)) \cdot p^2, \\ \mathit{Cost}(\mathit{Southlist}) &\sim 4 \cdot p^3 + N(\mathit{Southlist}(b)) \cdot p^2, \end{aligned} \tag{55}$$

$$\begin{aligned} Cost(Eastlist) &\sim 4 \cdot p^3 + N(Eastlist(b)) \cdot p^2, \\ Cost(Westlist) &\sim 4 \cdot p^3 + N(Westlist(b)) \cdot p^2, \end{aligned}$$

respectively. Combining (55) with the fact that the maximum total number of boxes in the interaction list is 189, we obtain

$$Cost(\mathcal{T}_{ML}) \sim 20 \cdot p^3 + 189 \cdot p^2. \tag{56}$$

Remark 3.10 The procedure of the preceding section has been further accelerated. First, symmetry considerations can be used to reduce number of translations per box from 189 to 40 without any loss of precision. We refer the reader to [12] for details. Second, while the expansions (5) and (8) are expressed in terms of spherical harmonics, they are being used to represent potentials inside or outside of regions that are cubic in shape. Clearly, spherical harmonics are not an optimal basis for this purpose. Special-purpose harmonics have been developed for the representation of potentials in such regions; they have been incorporated in our implementation and the timings presented in Section 6 below reflect this additional improvement. The procedure itself is fairly involved, and will be reported at a later date [6].

4 The non-adaptive FMM

In this section, we describe the non-adaptive FMM algorithm of [12], combining the factorization (54) of the translation operator $\mathcal{T}_{ML}$ with the factorizations (36), (38) of the operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$. We present it here as a reference for the subsequent adaptive procedure. For details, the reader is referred to the original paper [12].

In the FMM, the set of all boxes at level l is denoted by B_l, with B_0 consisting of the computational box itself. With each box b, we associate fourteen expansions about its center.

- A *multipole expansion* Φ_b of the form (5) represents the potential generated by charges contained inside b; it is valid in $\mathbf{R}^3 \setminus Coll(b)$.
- A *local expansion* Ψ_b of the form (8) represents the potential generated by all charges outside $Coll(b)$; it is valid inside box b.
- Six *outgoing exponential expansions* W_b^{Up}, W_b^{Down}, W_b^{North}, W_b^{South}, W_b^{East}, and W_b^{West} of the form (39), representing the potential generated by all charges located in b and valid in $Uplist(b)$, $Downlist(b)$, $Northlist(b)$, $Southlist(b)$, $Eastlist(b)$, and $Westlist(b)$, respectively.
- Six *incoming exponential expansions* V_b^{Up} V_b^{Down}, V_b^{North}, V_b^{South}, V_b^{East}, and V_b^{West} of the form (39), representing the potential inside b generated by all charges located in $Downlist(b)$, $Uplist(b)$, $Southlist(b)$, $Northlist(b)$, $Westlist(b)$, and $Eastlist(b)$, respectively.

NON-ADAPTIVE FMM ALGORITHM

Initialization

Comment [Choose number of refinement levels $\mathbf{NLEV} \approx \log_8 N$, and the order p of the multipole expansions. The number of boxes at the finest level is then $8^{\mathbf{NLEV}}$, and the average number of particles per box is $s = N/(8^{\mathbf{NLEV}})$. Denote the set of all boxes at level l by B_l.]

Upward Pass

Step 1

Do for each box $b \in B_{\mathbf{NLEV}}$,
  Form multipole expansion Φ_b of potential field due to
  particles in box b at b's center, via Theorem 2.1.
End do

Step 2

Do for levels $l = \mathbf{NLEV} - 1, ..., 2$,
  Do for each box $b \in B_l$,
    Form multipole expansion Φ_b about the center of b by
    merging expansions from its eight children via Theorem 2.3.
    (In applying $\mathcal{T}_{MM}$, use the factorization of Observation 3.3.)
  End do
End do

Downward Pass

Initialization

Set $\Psi_b = (0, 0, ..., 0)$ for all boxes.

Step 3A

Do for levels $l = 2, ..., \mathbf{NLEV}$,
  Do for each box $b \in B_l$,
    Form local expansion Ψ_b about the center of b by
    using Theorem 2.5 to shift the local expansion of b's parent to b.
    (In applying $\mathcal{T}_{LL}$, use the factorization of Observation 3.3.)
  End do

Step 3B

Do for $Dir = Up, Down, North, South, East, West$,
  Do for each box $b \in B_l$,
    Convert the multipole expansion Φ_b to the
    "outgoing" exponential W_b^{Dir}, via Theorem 3.3.
    Do for each box $c \in Dir - list(b)$,
      Translate W_b^{Dir} from b to c via Theorem 3.2 and add to V_c^{Dir}.
    End do
  End do
  Do for each box $c \in B_l$,
    Convert the incoming exponential V_c^{Dir} to the
    local expansion Ψ_c, via Theorem 3.4.
  End do
End do
End do

Step 4

Do for each box $b \in B_{\mathbf{NLEV}}$,
  For each particle in box b, evaluate Ψ_b at the particle position.
End do

Step 5

Do for each box $b \in B_{\mathbf{NLEV}}$,
  For each particle in box b,
  compute interactions with particles in b's colleagues directly.
End do

---

5 The adaptive FMM

The preceding algorithm is efficient for reasonably uniform distributions of particles, but its performance deteriorates significantly for non-uniform distributions. To remedy this situation, we construct an adaptive version of the scheme. Our strategy follows closely that used in [4] for the two dimensional case. Starting with the computational box, we build our structure recursively. If the box under consideration contains no charges, its existence is immediately forgotten. If it contains fewer than s charges (where s is an appropriately chosen positive integer), it is not subdivided further and considered *childless*. Otherwise, it is considered a *parent box* and subdivided into its eight children. The procedure is then repeated for each of the latter. As in the nonadaptive case, the set of all nonempty boxes at level l is denoted by B_l, with B_0 consisting of the computational box itself.

5.1 Adaptive lists

In order to describe the adaptive scheme, we will need the following additional lists.

Definition 5.1 *List 1* of a childless box b, denoted by $L_1(b)$, is defined to be the set consisting of b and all childless boxes adjacent to b. If b is a parent box, its List 1 is empty.

Definition 5.2 *List 2* of a box b, denoted by $L_2(b)$, is the set consisting of all children of the colleagues of b's parent that are well separated from b.

Definition 5.3 *List 3* of a childless box b, denoted by $L_3(b)$, is the set consisting of all descendents of b's colleagues that are not adjacent to b, but whose parent boxes are adjacent to b. If b is a parent box, its list 3 is empty.

Note that any box c in $L_3(b)$ is smaller than b and is separated from b by a distance not less than the side of c, and not greater than the side of b.

Definition 5.4 *List 4* of a box b, denoted by $L_4(b)$, consists of boxes c such that $b \in L_3(c)$; in other words, $c \in L_4(b)$ if and only if $b \in L_3(c)$.

Note that all boxes in $L_4(b)$ are childless and are larger than b.

Figure 4 shows the four lists for a box b in two dimensions. Of these, *List 1* and *List 2* have simple analogues in the non-adaptive algorithm of Section 4. Specifically, *List 1* of some finest level box b would consist of its colleagues, whose interactions will be accounted for directly. *List 2* of b would consist of boxes that are of the same size as b and are well separated: i.e., the interaction list of Definition 3.4. Lists 3 and 4 do not have analogues in the non-adaptive scheme.

$L_2(b)$ is subdivided further into *Uplist*(b), *Downlist*(b), *Northlist*(b), *Southlist*(b), *Eastlist*(b), and *Westlist*(b), by obvious analogy with Definition 3.5.

With each box b, we also associate fourteen expansions by analogy with those described in section 4. The only difference is that the ***multipole expansion*** Φ_b is valid in $\mathbf{R}^3 \setminus (L_1(b) \cup L_3(b))$. Similarly, the ***local expansion*** Ψ_b represents the potential inside b generated by all charges outside $L_1(b) \cup L_3(b)$.

ADAPTIVE FMM ALGORITHM

Initialization

Choose precision ε and the order of the multipole expansions p. Choose the maximum number s of charges allowed in a childless box. Define B_0 to be the smallest cube containing all sources (the computational domain).

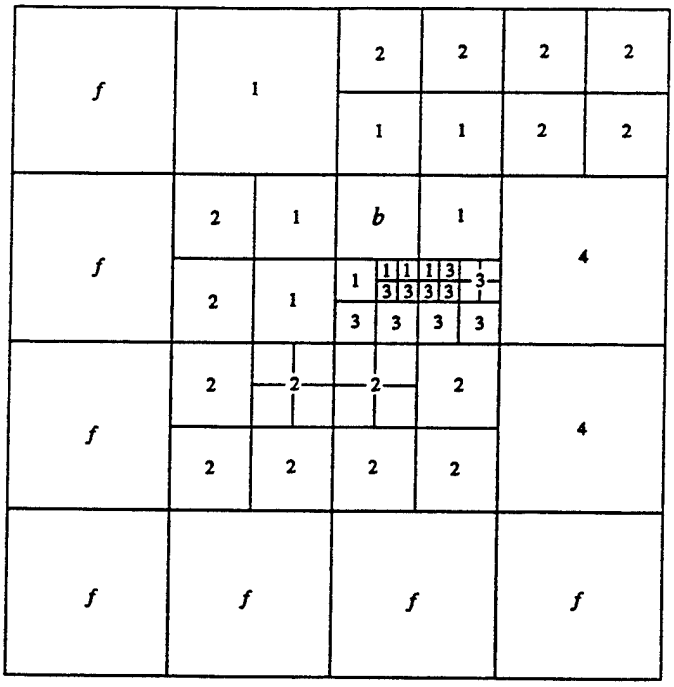

Figure 4: Lists 1-4 for box b

Build Tree Structure

Step 0

Do for levels $l = 0, 1, 2, ...$
  Do for each box $b \in B_l$
    If b contains more than s charges **then**
      Divide b into eight child boxes. Ignore empty children
      and add the nonempty child boxes to B_{l+1}.
    End if
  End do
End do

Comment [Denote the greatest refinement level obtained above by **NLEV** and the total number of boxes created as **NBOX**. Create the four lists for each box.]

Do for each box $b_i, i = 1, 2, ..., \mathbf{NBOX}$
  Create lists $L_1(b_i), L_2(b_i), L_3(b_i), L_4(b_i)$.
  Split $L_2(b_i)$ into *Up, Down, North, South, East, West* lists.
End do

Upward Pass

Comment [During the upward pass, a pth-order multipole expansion is formed for each box b about its center, representing the potential in $\mathbf{R}^3 \setminus (L_1(b) \cup L_3(b))$ due to all charges in b.]

Step 1

Comment [For each childless box b, form a multipole expansion about its center from all charges in b.]

Do for each box $b_i, i = 1, 2, ..., \mathbf{NBOX}$
  If b_i is childless **then**
    Use Theorem 2.1 to form pth-order multipole expansion Φ_{b_i},
    representing the potential in $\mathbf{R}^3 \setminus (L_1(b) \cup L_3(b))$ due to all charges in b_i.
  End if
End do

Step 2

Comment [For each parent box, form a multipole expansion about its center by merging multipole expansions from its children.]

Do for levels $l = \mathbf{NLEV} - 1, \mathbf{NLEV} - 2, ..., 0$
  Do for each box $b \in B_l$
    If b is a parent box **then**
      Use the operator $\mathcal{T}_{MM}$ to merge multipole expansions from
      its children into Φ_b.
    End if
  End do
End do

Downward Pass

Comment [During the downward pass, a pth-order local expansion is generated for each box b about its center, representing the potential in b due to all charges outside $(L_1(b) \cup L_3(b))$.]

Step 3

Comment [For each box b, add to its local expansion the contribution due to charges in $L_4(b)$.]

Do for each box $b_i, i = 1, 2, \cdots, \mathbf{NBOX}$
  Do for each box $c \in L_4(b_i)$
    If the number of charges in $b_i \leq p^2$ **then**

Comment [The number of charges in b_i is small. It is faster to use direct calculation than to generate the contribution to the local expansion Ψ_{b_i} due to charges in c; act accordingly.]

Calculate potential field at each particle point in b_i
directly from charges in c.
Else

Comment [The number of charges in b_i is large. It is faster to generate the contribution to the local expansion Ψ_{b_i} due to charges in c than to use direct calculation; act accordingly.]

Generate a local expansion at b_i's center due to
charges in c, and add to Ψ_{b_i}.
End if
End do
End do

Step 4

Comment [For each box b on level l with $l = 2, 3, ..., \mathbf{NLEV}$ and for each direction $Dir = Up, Down, North, South, East, West$, create from box b's multipole expansion the outgoing exponential W_b^{Dir} in direction Dir, using the operator $\mathcal{C}_{MX}$. Translate W_b^{Dir} to the center of each box $c \in Dirlist(b)$ using Corollary 3.2, and add the translated expansions to its incoming exponential expansion V_c^{Dir}.]

Do for levels $l = 2, 3, ..., \mathbf{NLEV}$
Do for $Dir = Up, Down, North, South, East, West$
Do for each box $b \in B_l$
Use the operator $\mathcal{C}_{MX}$ to convert multipole expansion
Φ_b into exponential W_b^{Dir}.
Do for each box $c \in Dirlist(b)$
Translate the outgoing exponential expansion W_b^{Dir} to the center of box c
using the diagonal translation operator $\mathcal{D}_{XX}$, and add the translated
expansion to the incoming exponential expansion V_c^{Dir}.
End do
End do

Comment [For each box c on level l, convert the exponential expansion V_c^{Dir} into a local expansion and add it to Ψ_c.]

Do for each box $c \in B_l$
Use the operator $\mathcal{C}_{XL}$ to convert the exponential expansion V_c^{Dir}
into a local expansion, and add it to Ψ_c.
End do
End do
End do

Step 5

Comment [For each parent box b, shift the center of its local expansion to its children.]

Do for each box $b_i, i = 1, 2, \cdots,$ **NBOX**
  If b_i is a parent box **then**
    Use the operator $\mathcal{T}_{LL}$ to shift the local expansion Ψ_{b_i} to the centers of its children, and add the translated expansions to children's local expansions.
  End if
End do

Evaluation of Potentials

Step 6

Comment [Include contribution to potential from local expansion at leaf nodes.]

Do for each box $b_i, i = 1, 2, ...,$ **NBOX**
  If b_i is childless **then**
    Calculate the potential at each charge in b_i from the local expansion Ψ_{b_i}.
  End if
End do

Step 7

Comment [Include contribution from direct interactions.]

Do for each box $b_i, i = 1, 2, ...,$ **NBOX**
  If b_i is childless **then**
    Calculate the potential at each charge in b_i
    directly due to all charges in $L_1(b_i)$.
  End if
End do

Step 8

Comment [For each childless box b, evaluate the potential due to all charges in $L_3(b)$.]

Do for each box $b_i, i = 1, 2, ...,$ **NBOX**
  If b_i is childless **then**
    Do for each box $c \in L_3(b_i)$
      If the number of charges in $c \leq p^2$ **then**

Comment [The number of charges in c is small. It is faster to use direct calculation than to evaluate the multipole expansion Φ_c; act accordingly.]

Calculate the potential at each charge in b_i
directly from charges in c.

Else

Comment [The number of charges in c is large. It is faster to evaluate the expansion Φ_c than to use direct calculation; act accordingly.]

Calculate the potential at each charge in b_i
from multipole expansion Φ_c.

End if

End do

End if

End do

Remark 5.1 *Step 3* in the above algorithm could be simplified without increasing the asymptotic CPU time estimate of the latter. Specifically, we could always generate the contribution to the local expansion Ψ_b due to charges in c, even when the number of charges in c is small. However, the actual computation time would increase somewhat. A similar observation can be made about *Step 8* of the above algorithm.

Remark 5.2 In the actual implementation of the adaptive algorithm, we have introduced several minor modifications, designed primarily to reduce the memory requirements of the scheme. In particular, *Steps 3, 4*, and *5* of the downward pass have been combined to eliminate some of the intermediate storage.

5.2 Complexity Analysis and Comparison with Tree Codes

The cost of the FMM algorithm of this paper (like the cost of older schemes of this type) can be separated into two parts. The first part concerns the construction of the data structure (*Step 0*); the second part concerns the calculation of the potentials.

If N denotes the total number of particles in the system, the CPU time estimate for the first part is $O(N \log N)$ in the general case and $O(N)$ for reasonably uniform distributions of particles, where "bin sorting" can be used instead of the recursive procedure outlined above. The CPU time requirements for the second part are $O(N)$ in all cases. In practice, however, the first part uses a negligible proportion of the total CPU time.

There has been some confusion in the literature concerning computational complexity, partly because of an erroneous proof in the original paper [4] addressing the two dimensional case. A correct proof can be found in [17], under very general assumptions about the distribution of charges. We omit the detailed analysis of the asymptotic time and storage estimates for the algorithm of this paper since it does not differ materially from that in [17]. For reasonably

uniform distributions, it is easy to see that the asymptotic cost of the nonadaptive algorithm is approximately

$$27Ns + 2Np^2 + 189\frac{N}{s}p^2 + 20\frac{N}{s}p^3,$$

where s is the number of charges per box at the finest level. The first term comes from direct interactions with colleagues, the second comes from forming and evaluating multipole and local expansions at the finest level, and the last two come from multipole-to-local translations, as shown in (56). Using symmetry considerations, it is possible to reduce the factor 189 to 40 (see Remark 3.10 above). Setting $s \approx p^{3/2}$, we see that the work required by the nonadaptive FMM is of the order

$$O(Np^{3/2}).$$

Similarly, the storage costs are of the order

$$O(\frac{N}{s}p^2) \sim O(Np^{3/2}).$$

In the adaptive case, precise estimates are more involved, but the reader will note in the numerical examples below that both CPU times and storage requirements are at a maximum for the most homogenous distributions.

A second area where there has been some confusion concerns comparisons of the FMM with what are generally known as "tree codes." These were introduced independently of the FMM by Barnes and Hut [2]. (A related scheme by Appel [1] is more like the FMM than like a tree code.) In tree codes, all interactions are computed by either direct calculation or by evaluation of a multipole expansion for a source box at a well-separated target position. Within the FMM, however, one has four options for a source box b and a target box c:

1. compute interactions directly,
2. evaluate the multipole expansion for b at individual targets in c directly,
3. convert the field due to each source in b to a local expansion in c (which is later evaluated),
4. convert the multipole expansion in b to a local expansion in c (which is later evaluated).

A properly implemented FMM always selects the least expensive option (which is trivial to choose); thus, it is always more efficient than a tree code. We omitted this decision analysis in our original descriptions of the FMM [10, 11, 18] in order to focus on the central result, which is option 4 above. It is this option which reduces the cost to $O(N)$. It is easy to see that options 2 and 3 are appropriate only in Steps 3 and 8 above, when considering Lists 3 and 4. The analogues of Steps 3 and 8 here are Stages 5 and 6 in [4].

6 Numerical Results

The algorithm described in Section 5 has been implemented in Fortran 77, and numerical experiments have been carried out for a variety of charge distributions using a Sun UltraSPARC workstation with a CPU clock rate of 167 MHz. The results of our experiments are summarized in Tables 1-12, with all timings given in seconds.

In the first set of our experiments, the charges were distributed randomly but uniformly in the cube $[-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$; results are reported in Tables 1-3. In the second set, the charges were distributed randomly in the polar angles θ and ϕ on the surface of a sphere of radius 0.5, centered at the origin. Obviously, such a distribution is concentrated at the poles (Figure 5); results are reported in Tables 4-6. In the third set, the charges were distributed on the surface of a cylinder with height 1.0 and radius 0.05 (Figure 6); results are reported in Tables 7-9. In the final set of experiments, the charges were distributed on a complicated surface shown in Figure 7. The results for this configuration are reported in Tables 10-12. In all our experiments, the charge strengths were taken randomly from the interval $(-0.5, 0.5)$.

For each geometry, the numerical tests were performed with three-, six-, and nine-digit accuracy. For three-digit accuracy, the maximum number of charges allowed in a childless box was set to be 40. Corresponding numbers for six- and nine-digit accuracies are 100 and 180, respectively. The timings produced by the adaptive FMM algorithm were compared with those obtained by the direct calculation. Obviously, it was not practical to apply the direct scheme to large-scale ensembles of particles, due to excessive computation times. Thus, the direct algorithm was used to evaluate the potentials at the first 100 elements of the ensemble, and the resulting CPU time was extrapolated. Similarly, the accuracy of the algorithm was calculated at the first 100 particles via formula (57) below.

The tables are organized as follows.

1. The first column lists the number of charges used in the calculation.

2. The second column lists the number of levels used in the multipole hierarchy.

3. The third column lists the order of the multipole expansion used.

4. The fourth column lists the corresponding number of exponential basis functions.

5. The fifth column lists the amount of storage used by the adaptive FMM algorithm. In the three- and six-digit cases, we indicate the number of single precision (REAL*4) words used, while in the nine-digit case, we indicate the number of double precision (REAL*8) words used.

6. Columns six and seven contain the CPU times required by the adaptive FMM and the direct calculation, respectively. In the three- and six-digit cases, both the FMM and the direct calculations were performed in single precision; in the nine-digit case, both calculations were performed in double precision.

7. Column eight lists the L_2 norm of the error in the FMM approximation, which is computed via the formula

$$E = \left(\frac{\sum_{i=1}^{N} |\Phi(x_i) - \tilde{\Phi}(x_i)|^2}{\sum_{i=1}^{N} |\Phi(x_i)|^2} \right)^{1/2}, \tag{57}$$

where $\tilde{\Phi}(x_i)$ are potentials obtained by the FMM algorithm and $\Phi(x_i)$ are potentials computed by direct calculation in double precision.

Table 1: Timing results for the FMM for 3-digit of accuracy with charges uniformly distributed in a cube. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	4	2267	10	52	1359822	13.3	233	$7.9 \cdot 10^{-4}$
50000	4	4681	10	52	3365896	24.7	1483	$5.2 \cdot 10^{-4}$
200000	5	33749	10	52	24789948	158	24330	$8.4 \cdot 10^{-4}$
500000	5	37449	10	52	28835176	268	138380	$7.0 \cdot 10^{-4}$
1000000	6	48324	10	52	34798506	655	563900	$7.1 \cdot 10^{-4}$

Table 2: Timing results for the FMM for 6-digit of accuracy with charges uniformly distributed in a cube. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	3	585	19	258	1057852	15.9	233	$5.1 \cdot 10^{-7}$
50000	4	2065	19	258	3383488	69	1483	$2.8 \cdot 10^{-7}$
200000	4	4681	19	258	8220716	198	24330	$4.9 \cdot 10^{-7}$
500000	5	36665	19	258	64326704	586	138380	$4.4 \cdot 10^{-7}$
1000000	5	37449	19	258	66414780	1245	563900	$4.4 \cdot 10^{-7}$

The following observations can be made from these tables.

1. The application of the FMM to large scale three dimensional problems is within practical reach.
2. The actual CPU time required by the adaptive FMM algorithm grows approximately linearly with the number of particles N.
3. The algorithm breaks even with the direct calculation at about $N = 750$ for three-digit precision, $N = 1500$ for six-digit precision and $N = 2500$ for nine-digit precision.
4. The performance of the algorithm is quite insensitive to the distribution of charges.

Table 3: Timing results for the FMM for 9-digit of accuracy with charges uniformly distributed in a cube. Calculations were performed in double precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	3	585	29	670	2012453	34	296	$2.8 \cdot 10^{-10}$
50000	3	585	29	670	2012453	96	1920	$1.6 \cdot 10^{-10}$
200000	4	4681	29	670	16479203	385	30800	$1.6 \cdot 10^{-10}$
500000	4	4681	29	670	16479203	1219	192600	$1.2 \cdot 10^{-10}$

Table 4: Timing results for the FMM for 3-digit of accuracy with charges distributed on the surface of a sphere. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	7	1746	10	52	891080	8.7	233	$4.2 \cdot 10^{-4}$
50000	9	4757	10	52	2394568	21.6	1483	$3.6 \cdot 10^{-4}$
200000	11	18221	10	52	9126212	97	24330	$8.0 \cdot 10^{-4}$
500000	12	40717	10	52	20413944	224	138380	$6.4 \cdot 10^{-4}$
1000000	13	90139	10	52	45287934	473	563900	$5.5 \cdot 10^{-4}$

Table 5: Timing results for the FMM for 6-digit of accuracy with charges distributed on the surface of a sphere. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	6	624	19	258	1037742	16	233	$2.4 \cdot 10^{-7}$
50000	7	1774	19	258	2774248	40	1483	$2.7 \cdot 10^{-7}$
200000	9	6790	19	258	10365264	183	24330	$2.3 \cdot 10^{-7}$
500000	10	18897	19	258	28580428	529	138380	$4.3 \cdot 10^{-7}$
1000000	11	33289	19	258	50405060	926	563900	$2.9 \cdot 10^{-7}$

Table 6: Timing results for the FMM for 9-digit of accuracy with charges distributed on the surface of a sphere. Calculations were performed in double precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	5	429	29	670	1422805	33	296	$3.2 \cdot 10^{-11}$
50000	6	1091	29	670	3616209	98	1920	$8.1 \cdot 10^{-11}$
200000	8	4342	29	670	14394468	409	30800	$7.6 \cdot 10^{-11}$
500000	10	9009	29	670	29828865	1038	192600	$1.2 \cdot 10^{-10}$

Table 7: Timing results for the FMM for 3-digit of accuracy with charges distributed on the surface of a cylinder. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	6	1963	10	52	1013298	8.2	233	$2.7 \cdot 10^{-4}$
50000	7	4084	10	52	2014394	20.8	1483	$4.0 \cdot 10^{-4}$
200000	8	18795	10	52	9056494	93	24330	$5.1 \cdot 10^{-4}$
500000	9	31093	10	52	15409424	194	138380	$5.1 \cdot 10^{-4}$
1000000	9	101374	10	52	49326404	457	563900	$4.9 \cdot 10^{-4}$

Table 8: Timing results for the FMM for 6-digit of accuracy with charges distributed on the surface of a cylinder. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	5	505	19	258	868700	13.8	233	$2.5 \cdot 10^{-7}$
50000	6	2037	19	258	3180832	39	1483	$2.9 \cdot 10^{-7}$
200000	7	7001	19	258	10582852	143	24330	$5.6 \cdot 10^{-7}$
500000	8	19849	19	258	29654956	508	138380	$7.0 \cdot 10^{-7}$
1000000	8	29341	19	258	44253336	921	563900	$6.4 \cdot 10^{-7}$

Table 9: Timing results for the FMM for 9-digit of accuracy with charges distributed on the surface of a cylinder. Calculations were performed in double precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20000	5	505	29	670	1676098	30	296	$2.8 \cdot 10^{-11}$
50000	6	751	29	670	2478241	86	1920	$5.1 \cdot 10^{-11}$
200000	7	2515	29	670	8348058	341	30800	$8.2 \cdot 10^{-11}$
500000	7	7344	29	670	24250893	795	192600	$9.4 \cdot 10^{-11}$

Table 10: Timing results for the FMM for 3-digit of accuracy with charges distributed as in Fig. 7. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20880	7	1213	10	52	573996	6.7	243	$2.2 \cdot 10^{-4}$
51900	8	4184	10	52	1952046	17	1539	$2.7 \cdot 10^{-4}$
203280	9	15423	10	52	7204398	60	24730	$3.4 \cdot 10^{-4}$
503775	10	45837	10	52	21358082	164	141060	$3.3 \cdot 10^{-4}$
1007655	10	60427	10	52	28513092	282	568090	$2.9 \cdot 10^{-4}$

Table 11: Timing results for the FMM for 6-digit of accuracy with charges distributed as in Fig. 7. Calculations were performed in single precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20880	7	1038	19	258	1601028	17	243	$1.3 \cdot 10^{-7}$
51900	8	1403	19	258	2165338	40	1539	$9.8 \cdot 10^{-8}$
203280	9	4447	19	258	6697050	149	24730	$1.2 \cdot 10^{-7}$
503775	9	15307	19	258	22662792	323	141060	$2.6 \cdot 10^{-7}$
1007655	10	45784	19	258	67176488	714	568090	$2.0 \cdot 10^{-7}$

Table 12: Timing results for the FMM for 9-digit of accuracy with charges distributed as in Fig. 7. Calculations were performed in double precision.

N	Levels	Boxes	p	S_{exp}	Storage	T_{FMM}	T_{DIR}	Error
20880	6	574	29	670	1856177	46	309	$3.6 \cdot 10^{-12}$
51900	7	1191	29	670	3855741	101	2020	$1.1 \cdot 10^{-10}$
203280	8	3883	29	670	12577869	342	32050	$6.5 \cdot 10^{-12}$
503775	9	11499	29	670	37263647	896	193900	$1.0 \cdot 10^{-11}$

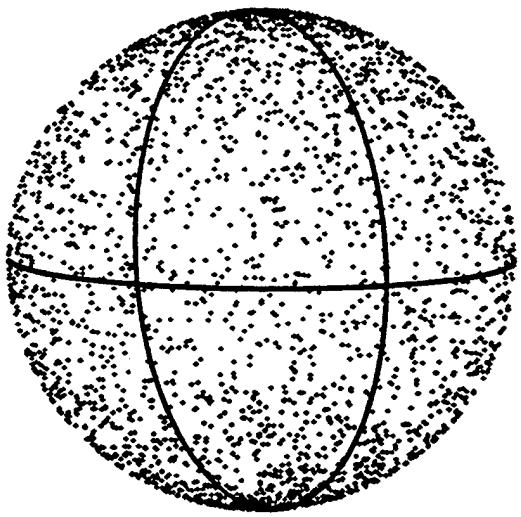

Figure 5: Charges distributed on the surface of a sphere.

Figure 6: Charges distributed on the surface of a cylinder.

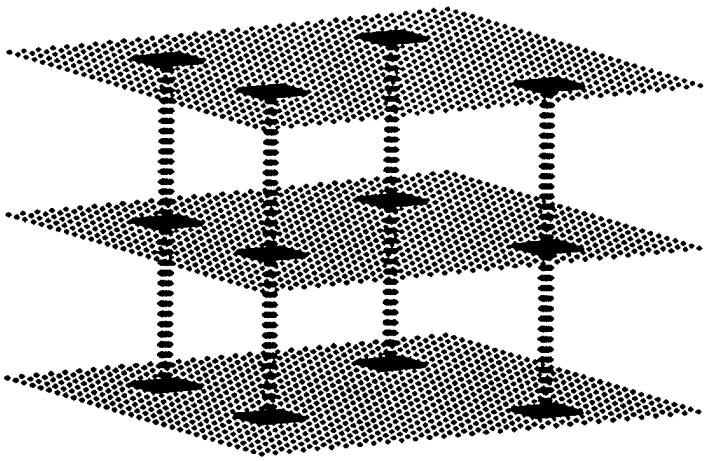

Figure 7: Charges distributed on a complicated object.

7 Generalizations and Conclusions

We have described an adaptive FMM for the Laplace equation based on a new diagonal form for translation operators acting on harmonic functions. It is related to the FMM for the high-frequency Helmholtz equation, in the sense that the latter is based on diagonal forms of translation operators for partial wave expansions [7, 19, 20].

The present scheme admits a number of extensions. The most straightforward ones are to the Helmholtz equation at low frequencies and to the Yukawa equation. The corresponding multipole expansions are well-known, and appropriate plane wave representations have been derived (see, for example, [13]).

From a more abstract perspective, it is worth noting that the main improvement made in this paper and in [12] over earlier FMMs is due to the use of one basis for representing the far field due to a collection of sources (spherical harmonics) and a separate basis for translating information between boxes in the FMM data structure (plane waves). The applicability of this approach is not limited to the Laplace and Helmholtz equations. We are currently in the process of constructing such optimal (or nearly optimal) bases for more general potentials, including those that do not satisfy a partial differential equation, but possess certain less stringent analytical properties. A forthcoming paper [8] describes such an algorithm for the square root of the Laplacian in two dimensions; further generalizations will be reported at a later date.

8 Appendix

The three tables in this Appendix contain the nodes and weights (in columns 2 and 3) needed for discretization of the outer integral in Lemma 2.8. Column 4 contains the number of discretization points needed in the inner integral, which we denote by M_k^d.

Table 13: Nodes, weights and M_k^3 for 3-digit accuracy.

k	*Node*	*Weight*	M_k^3
1	0.10934746769000	0.27107502662774	4
2	0.51769741015341	0.52769158843946	8
3	1.13306591611192	0.69151504413879	16
4	1.88135015110740	0.79834400406452	16
5	2.71785409601205	0.87164160121354	24
6	3.61650274907449	0.92643839116924	24
7	4.56271053303821	0.97294622259483	8
8	5.54900885348528	1.02413865844686	4

Table 14: Nodes, weights and M_k^6 for 6-digit accuracy.

k	$Node$	$Weight$	M_k^6
1	0.05599002531749	0.14239483712194	8
2	0.28485138101968	0.31017671029271	8
3	0.66535367065853	0.44557516683709	16
4	1.16667904805296	0.55303383994159	16
5	1.76443027413431	0.63944903363523	24
6	2.44029832236380	0.70997911214019	32
7	3.18032180991515	0.76828253949732	32
8	3.97371715777193	0.81713201141707	32
9	4.81216799410634	0.85872191623337	48
10	5.68932314511487	0.89480789582390	48
11	6.60040479444377	0.92680189417317	48
12	7.54190497469911	0.95586282708096	48
13	8.51136569298099	0.98299145008230	48
14	9.50723242759128	1.00913395385703	48
15	10.52874809650967	1.03531774600508	48
16	11.57587019602884	1.06318427913963	8
17	12.65078163968520	1.10232109521088	4

Table 15: Nodes, weights and M_k^9 for 9-digit accuracy.

k	$Node$	$Weight$	M_k^9
1	0.03705701953816	0.09473396337900	8
2	0.19219683859955	0.21384206006426	16
3	0.46045971214897	0.32031528543989	16
4	0.82805130101422	0.41254929390710	16
5	1.28121229944787	0.49176691815621	24
6	1.80792019276297	0.55998309037174	32
7	2.39814728074333	0.61909314036708	32
8	3.04359012306582	0.67064351982741	32
9	3.73732742924096	0.71586567032066	48
10	4.47354768940212	0.75576118553096	48
11	5.24735518169467	0.79116885492295	48
12	6.05462948620944	0.82280556212477	64
13	6.89191648795972	0.85129012269433	64
14	7.75633860708838	0.87715909928110	64
15	8.64551915195994	0.90087981520398	64
16	9.55751929613924	0.92286282936149	72
17	10.49078760616705	0.94347471535979	72
18	11.44412262341269	0.96305166489156	80
19	12.41664955395045	0.98191478773737	80
20	13.40781311788324	1.00038891281291	88
21	14.41739038894472	1.01882849188686	88
22	15.44553016867884	1.03765781507554	88
23	16.49282861241170	1.05744113465683	88
24	17.56045648926099	1.07903824697122	72
25	18.65046484106274	1.10434337868208	32
26	19.76847686619416	1.14488166506896	4

References

[1] A. W. APPEL (1985), "An efficient program for many-body simulation", *SIAM J. Sci. Stat. Comput.* **6**, 85–103.

[2] J. BARNES AND P. HUT (1986), "A hierarchical $O(N \log N)$ force-calculation algorithm", *Nature* **324**, 446–449.

[3] L. C. BIEDENHARN AND J. D. LOUCK (1981), *Angular Momentum in Quantum Physics : Theory and Application*, Addison-Wesley, Reading, Mass.

[4] J. CARRIER, L. GREENGARD, AND V. ROKHLIN (1988), "A fast adaptive multipole algorithm for particle simulations", *SIAM J. Sci. Statist. Comput.* **9**, 669–686.

[5] H. CHENG (1995). *Fast, accurate methods for the evaluation of harmonic fields in composite materials*, Ph.D. thesis, New York University.

[6] H. CHENG AND V. ROKHLIN (1998). "Compression of Translation operators in Fast Multipole Algorithms in Three Dimensions", in preparation.

[7] R. COIFMAN, V. ROKHLIN, AND S. WANDZURA (1993), "The fast multipole method for the wave equation: a pedestrian prescription", *IEEE Antennas and Propagation Mag.* **35**, 7.

[8] Z. GIMBUTAS, L. GREENGARD, AND M. MINION (1998), "A fast multipole method for the square root of the Laplacian", in preparation.

[9] L. GREENGARD (1988), *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, Mass.

[10] L. GREENGARD AND V. ROKHLIN (1987), "A fast algorithm for particle simulations", *J. Comput. Phys.* **73**, 325–348.

[11] L. GREENGARD AND V. ROKHLIN (1988a). "Rapid evaluation of potential fields in three dimensions", in *Vortex Methods*, C. Anderson and C. Greengard (eds.), Lecture Notes in Mathematics, vol. 1360, Springer-Verlag, 121.

[12] L. GREENGARD AND V. ROKHLIN (1997). "A new version of the fast multipole method for the Laplace equation in three dimensions", *Acta Numerica* 6, 229-269.

[13] J. HUANG, L. GREENGARD, V. ROKHLIN, AND S. WANDZURA, "Accelerating Fast Multipole Methods for Low Frequency Scattering" CMCL Report 1998-003, New York University (to appear in IEEE Computational Science and Engineering Magazine).

[14] J. D. JACKSON (1975), *Classical Electrodynamics*, Wiley, New York.

[15] O. D. KELLOGG (1953), *Foundations of Potential Theory*, Dover, New York.

[16] MORSE AND FESHBACH (1953), *Methods of Theoretical Physics*, McGraw-Hill, New York.

[17] K. NABORS, F. T. KORSMEYER, F. T. LEIGHTON, AND J. WHITE (1994). "Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory", *SIAM J. Sci. Stat. Comput.* **15**, 714.

[18] V. ROKHLIN (1985), "Rapid solution of integral equations of classical potential theory", *J. Comput. Phys.* **60**, 187–207.

[19] V. ROKHLIN (1990), "Rapid solution of integral equations of scattering theory in two dimensions", *J. Comput. Phys.* **86**, 414–439.

[20] V. ROKHLIN (1993), "Diagonal forms of translation operators for the Helmholtz equation in three dimensions", *Appl. and Comput. Harmonic Analysis* **1**, 82–93.

[21] P. R. WALLACE (1984). *Mathematical Analysis of Physical Problems*, Dover, New York.

[22] C.A. WHITE AND M. HEAD-GORDON (1996). "Rotating around the quartic angular momentum barrier in Fast Multipole Method calculation", *J. Chem. Phys.* **105**, 5061.

[23] N. YARVIN AND V. ROKHLIN (1996). "Generalized Gaussian quadratures and singular value decompositions of integral operators", *Department of Computer Science Research Report 1109, Yale University.*